

AD-A086 126

HOUSTON UNIV TX DEPT OF ELECTRICAL ENGINEERING

F/G 1/3

THE REMOTE LINK UNIT: APPLICATIONS TO THE DESIGN-FOR-REPAIR NET--ETC(III)

APR 80 C J TAVORA, M A SMITHER

F33615-78-C-1634

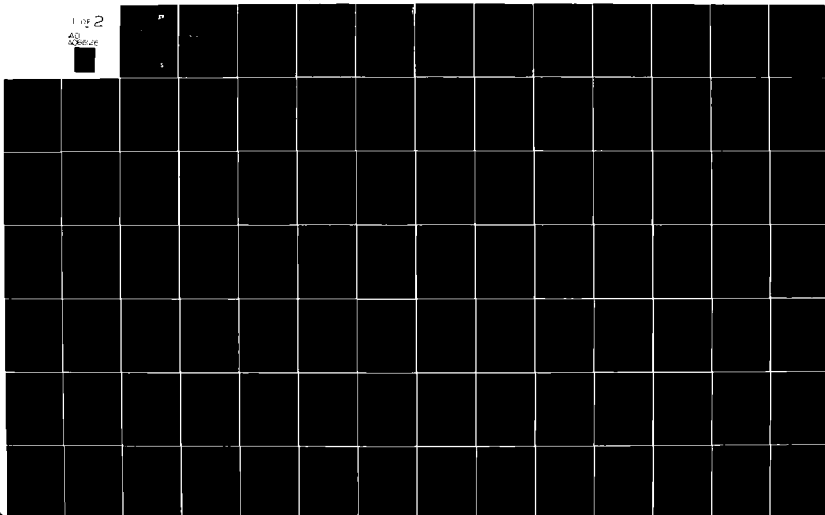
UNCLASSIFIED

AFWAL-TR-80-1033

HL

1 of 2

AD-A086 126



AFWAL-TR-80-1033

**LEVEL**



ADA 086126

## THE REMOTE LINK UNIT : APPLICATIONS TO THE DESIGN FOR REPAIR METHODOLOGY PROGRAM

ELECTRICAL ENGINEERING DEPARTMENT  
UNIVERSITY OF HOUSTON  
4800 CALHOUN BOULEVARD  
HOUSTON, TEXAS 77004

APRIL 1980

TECHNICAL REPORT AFWAL-TR-80-1033  
Final Report for August 15, 1979-March 15, 1980

DDC FILE COPY

Approved for public release; distribution unlimited

AVIONICS LABORATORY  
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES  
AIR FORCE SYSTEMS COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

DTIC  
ELECTE  
JUL 3 1980  
S A D

80 7 1 118

NOTICE

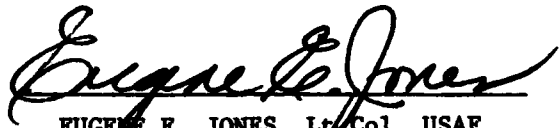
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

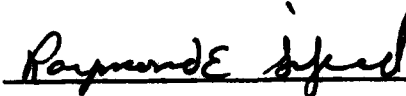


PHILIP C. GOLDMAN  
Project Engineer  
System Design Group



EUGENE E. JONES, Lt. Col, USAF  
Chief  
Avionic Systems Engineering Branch

FOR THE COMMANDER



RAYMOND E. SIFERD, Colonel, USAF  
Chief  
System Avionics Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFMIL/AAAA, W-PAGE, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWAL-TR-80-1033	2. GOVT ACCESSION NO. AD-A086 136	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Remote Link Unit: Applications to the Design-for-Repair Methodology Program -		5. TYPE OF REPORT & PERIOD COVERED Final Report Aug. 15, 1979-March 15, 1980
7. AUTHOR(s) C. J. Tavora M. A. Smither		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electrical Engineering Department University of Houston 4800 Calhoun Blvd., Houston, TX 77004		8. CONTRACT OR GRANT NUMBER(s) F33615-78-C-1634 Mod. P00002
11. CONTROLLING OFFICE NAME AND ADDRESS Avionics Laboratory AF Wright Aeronautical Laboratories (AFSC) Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2003-01-19
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE April 1980
		13. NUMBER OF PAGES 110
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Design-for-repair, Maintenance, Testability, Electronic Nameplate, Remote Link Unit, Remote Terminal, Distributed Avionics, Fault Monitoring, Fault recording, Automated maintenance, Fault tolerance.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This represents the results of a study on the application of the Remote Link Unit (RLU) and its design concepts to the Design-for-Repair Methodology Pro- gram. The report focuses on the incorporation of automated maintenance and fault tolerance in avionic systems. The major components used to implement the proposed design concepts are intelligent universal interface modules and electronic nameplates. A phased implementation plan for the integration of RLU concepts in the design of distributed avionic systems is also described.		

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE  
1 JAN 73

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## FOREWORD

This report was prepared by the University of Houston, on modification P00002 of Air Force Contract No. F33615-78-C-1634 entitled "The Remote Link Unit: An Advanced Remote Terminal for MIL-STD-1553B."

The contract was administered under the direction of the Systems Avionics Division of the Avionics Laboratory. Mr. Phil Goldman of the System Design Group was the project engineer. Technical support to the study was provided by Ms. Diane Summers, Technical Manager of the System Evaluation Group, and Lt. Donald Lowdermilk.

The Principal Investigator and Program Manager for this contract was Dr. Carlos J. Tavora. Dr. Miles A. Smither was the project Co-Investigator. Mr. Michael E. Brandt provided editing assistance in the preparation of this report.

The technical and administrative support provided by Lt. Col. Eugene Jones and Ms. Diane Summers was a significant factor in the success of this study.

Decision For	
Approved	<input checked="checked" type="checkbox"/>
Not Approved	<input type="checkbox"/>
Not Recommended	<input type="checkbox"/>
Indefinite	<input type="checkbox"/>
By _____	
Date _____	
Approved by _____	
Date _____	
Dist	Available for special
A	

## TABLE OF CONTENTS

SECTION	PAGE
I. INTRODUCTION. . . . .	1
1.1 STUDY OBJECTIVES. . . . .	1
1.2 THE RLU CONCEPT . . . . .	2
1.3 RLU SUPPORT OF DESIGN-FOR-REPAIR. . . . .	3
1.4 REPORT ORGANIZATION . . . . .	5
II. STATEMENT OF THE PROBLEM. . . . .	6
2.1 SYSTEM PERSPECTIVES OF DIFFERENT PERSONNEL. .11	
2.2 THE COMPLETE INTERFACE. . . . .	19
III. DESIGN FOR MAINTENANCE AND REPAIR . . . . .	24
3.1 FAILURE DETECTION AND REPORTING . . . . .	25
3.2 FAILURE ISOLATION AND REPAIR. . . . .	30
3.3 REPAIR VERIFICATION AND CALIBRATION . . . . .	43
IV. DESIGN FOR FAULT TOLERANCE. . . . .	48
4.1 SYSTEM DESIGN . . . . .	49
4.2 RLU USE IN FAULT TOLERANT SYSTEMS . . . . .	52
4.3 SUMMARY . . . . .	62
V. MAINTENANCE PROGRAMS AND DATA STORAGE . . . . .	65
5.1 SOFTWARE FOR MAINTENANCE AND FAULT TOLERANCE . . . . .	65
5.2 DATA STORAGE FOR MAINTENANCE. . . . .	71
5.3 PROGRAMMING LANGUAGES . . . . .	73

## TABLE OF CONTENTS (CON'T.)

SECTION	PAGE
VI. IMPLEMENTATION PLAN . . . . .	.75
6.1 PHASE I - SUBSYSTEM INTERFACE STANDARDIZATION . . . . .	.76
6.2 PHASE II - RLU INTEGRATION. . . . .	.78
6.3 PHASE III - AUTOMATED SYSTEM MAINTENANCE. .	.79
6.4 IMPLEMENTATION PLAN SUMMARY . . . . .	.81
VII. SUBSYSTEM DESIGN SPECIFICATIONS . . . . .	.83
7.1 INTERFACE SPECIFICATIONS. . . . .	.84
7.2 OPERATIONAL PERFORMANCE . . . . .	.85
7.3 TESTING SPECIFICATIONS. . . . .	.85
7.4 PROGRAM SPECIFICATIONS. . . . .	.86
7.5 NAMEPLATE SPECIFICATIONS. . . . .	.87
APPENDIX A - THE REMOTE LINK UNIT-AN ADVANCED REMOTE TERMINAL CONCEPT. . . . .	.89
APPENDIX B - RETROFIT USE OF THE RLU . . . . .	.95
REFERENCES . . . . .	110

## LIST OF ILLUSTRATIONS

FIGURE		PAGE
1	Architecture of an Avionics System Consisting of Autonomous Subsystem . . . . .	7
2	Architecture of an Integrated Digital Avionics System . . . . .	8
3	Conceptual View of an Avionics System Consisting of Autonomous Subsystems . . . .	12
4	Conceptual Views of an Integrated Digital Avionics System by Distinct Personnel . . .	14
5	The Complete Interface as a Delimiter of System and Subsystem Design . . . . .	22
6	RLU-Based Avionics System . . . . .	26
7	LM Controllable Test Example . . . . .	29
8	Reports Maintained by an RLU-Based Avionic System. . . . .	34
9	RLU Maintenance Flowchart - Organizational Level Repair. . . . .	42
10	Routine Functional Confirmation Test Configuration . . . . .	44
11	Calibration Cycle Adjustment Example . . . . .	47
12	Two Approaches to Fault Tolerant Design. . . .	50
13	System Reliability for Single Channel (p), Active Redundancy (Rar), and Standby Redundancy (Rsr) Cases. . . . .	53
14	RLU Use in Active Redundancy Fault Tolerant Design. . . . .	55
15	RLU Use in Standby Redundancy Fault Tolerant Design. . . . .	56
16	Redundant System-Level Data Paths. . . . .	59
17	Selective Redundancy at the LM/LRU Interface .	61

FIGURE		PAGE
18	Distributed Processing with Functional Clustering. . . . .	63
19	Storage and Loading of System and Subsystem Application Programs. . . . .	66
20	Flow of Control and Data Among Maintenance and Fault Tolerance Programs. . . . .	68
21	Hierarchical Structure for Storage of Malfunction Data. . . . .	72
22	Design-for-Repair System Architecture. . . . .	80
23	Time Phased Implementation of RLU Based Avionic Systems . . . . .	82

LIST OF TABLES

TABLE		PAGE
1	FEATURES OF A COMPLETE INTERFACE. . . . .	20
2	LEVEL 1 REPORT DATA BASE. . . . .	35
3	LEVEL 2 REPORT DATA BASE. . . . .	37
4	LEVEL 3 REPORT DATA BASE. . . . .	39
5	MAINTENANCE AND FAULT TOLERANCE SOFTWARE. . .	69

## LIST OF ABBREVIATIONS AND ACRONYMS

AGE	Aerospace Ground Equipment
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DAIS	Digital Avionics Information System
DRCD	Design-for-Repair Concept Definition
ICA	Interface Configuration Adapter
LM	Link Module
LMG	Link Manager
LRU	Line Replaceable Unit
MCF	Military Computer Family
MTBF	Mean Time Between Failures
NIC	Nameplate Interface Controller
NP	Electronic Nameplate
Q.A.	Quality Assurance
RF	Radio Frequency
RLU	Remote Link Unit
RT	Remote Terminal
RTU	Remote Terminal Unit
SDC	Subsystem Data Channel
SIC	Subsystem Information Channel
SRU	Shop Replaceable Unit
SS	Subsystem

## SECTION I

### INTRODUCTION

This document represents the results of a study concerning applications of the Remote Link Unit (RLU) to the Design-For-Repair methodology program. The RLU is a new design concept for remote terminals which incorporates ideas that may provide avionic systems with increased reliability, greater standardization, and automated maintenance and repair. This study is being conducted for the Avionics Laboratory under contract number F33615-78-C-1634.

#### 1.1 STUDY OBJECTIVES

This report addresses the following study objectives established in compliance with the statement of work for the above contract:

1. Analysis of the maintenance requirements of present avionic architectures which utilize remote terminals.
2. Evaluation of the electronic nameplate's potential to support automated maintenance and to eliminate or reduce maintenance paperwork.
3. Development of techniques to utilize RLU capability to support fault detection, fault isolation, fault recording, fault tolerance, and automated maintenance.

4. Identification of RLU software requirements for RLU support of fault tolerance and maintenance.
5. Development of an implementation plan that provides for a systematic incorporation of RLU concepts into the design of avionic systems.
6. Preparation of a final report that provides a broad perspective of the design-for-repair problem, of the possible solutions, and the manner in which they should be implemented.

## 1.2 THE RLU CONCEPT

The concept of a Remote Link Unit (RLU) is essentially an evolution of, or an expansion upon the concept of a remote terminal unit. The RLU provides a complete and direct interface between a CPU and remote subsystems. The RLU has universal interface modules (referred to as link modules) which are able to identify interfaced subsystems and to configure data and timing signals to match subsystem requirements. The subsystem identification and interface requirements are provided by an electronic nameplate which is interrogated by the link module. An additional feature of the electronic nameplate is that it will store programs for subsystem handling, engineering unit conversion, and subsystem calibration. These programs, when uploaded to the link module, will make details of the subsystem transparent

to CPU's. Subsystems interfaced through RLU's may be relocated or substituted without requiring changes in CPU software for correcting subsystem addresses or conversion constants. The electronic nameplate will also simplify inventory control, automatic calibration, and maintenance of subsystems. A complete description of the major features of the RLU may be found in the report "Remote Link Unit Functional Design: An Advanced Remote Terminal for MIL-STD-1553B,"[1]. An expanded description of the RLU concept is presented in Appendix A.

### 1.3 RLU SUPPORT OF DESIGN-FOR-REPAIR

The RLU concept provides a framework for the development of techniques which are supportive of the design-for-repair program. The following RLU based concepts are considered to be the most important for use in the design-for-repair effort:

1. Automation of maintenance through a central maintenance processing computer that communicates with the aircraft avionic system to retrieve and analyze the failures which occurred during a mission, to run diagnostic tests, and to prepare a service job order.
2. Reduce technical training required of avionics maintenance personnel by providing the RLU with an

interactive maintenance dialogue which facilitates testing and isolating LRU faults with the aid of a portable CRT terminal.

3. Separation between system and subsystem design with a well-defined information oriented interface which is independent of subsystem hardware peculiarities.
4. Generalization of the electronic nameplate concept to provide hierarchically distributed storage of recorded system failures (to be compatible with automated maintenance).
5. Development of a universal interface module (link module) to reduce logistics requirements.
6. A link module, with a processor which utilizes a machine-independent interpretive language, for use by subsystem designers to supplement subsystem hardware.
7. Improved transfer of technology by requiring subsystem designers to provide data conversion programs, diagnostic programs, and quality assurance programs for the subsystem.
8. Implementation of fault monitoring and recording as an integral function of the link module data conversion programs.
9. Implementation of a hierarchical fault tolerant architecture in which the RLU provides back up

processing functions at a level intermediate to subsystems and CPU's.

#### 1.4 REPORT ORGANIZATION

This report is divided into seven sections and two appendices. Section 2 describes the purposes of the present study through an analysis of the problems that led to the study. This section will attempt to pinpoint the limitations of present avionic systems and show how the concept of an RLU can eliminate those limitations. Section 3 describes how the RLU is utilized for the complete automation of avionic systems maintenance. Section 4 describes how the RLU can contribute to the design of fault tolerant avionic systems. Section 5 describes the RLU software and data storage requirements for support of fault tolerance and maintenance. Section 6 provides an implementation plan for the incorporation of RLU's into avionic systems. Section 7 outlines the specifications required to provide subsystem compatibility with automated maintenance. Appendix A provides a conceptual description of the RLU with particular attention to its components, structure, and operation. Appendix B is a description of how the RLU can be retrofitted into existing avionic systems to facilitate maintenance and improve fault tolerance.

## SECTION II

### STATEMENT OF THE PROBLEM

The design philosophy of avionic systems has drastically changed in the last ten years. One of the major causes for this change has been the development of microprocessors allowing the increased use of software in avionic systems.

Present avionic information systems may be classified into one of two categories: autonomous subsystems or integrated digital systems. The organization of an avionic system consisting of autonomous subsystems is depicted in Figure 1. Each individual subsystem contains all its unique sensors, signal conditioning, processing and display hardware required to implement its function. This type of system architecture is straightforward and its design can be performed by independent firms, with little interaction required among different firms involved in the design of other subsystems.

Due to increased usage of digital processing in most subsystems there has been strong technical and economic pressures to centralize subsystem processing in a general purpose CPU that performs all the processing required by the digital information system. The architecture of an integrated information system is illustrated in Figure 2. The key element of an integrated digital architecture is the

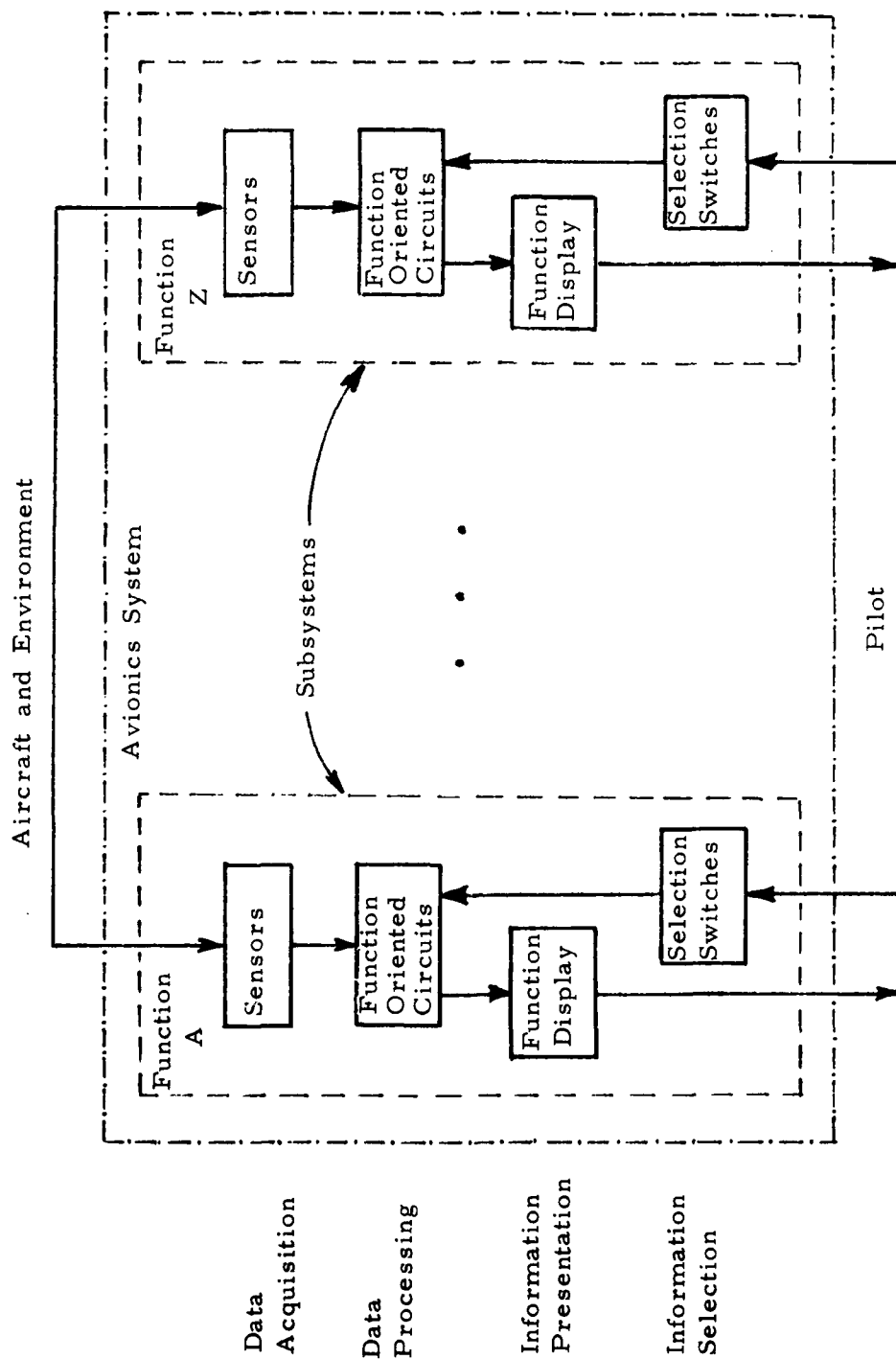


Figure 1 Architecture of an Avionics System Consisting of Autonomous Subsystems

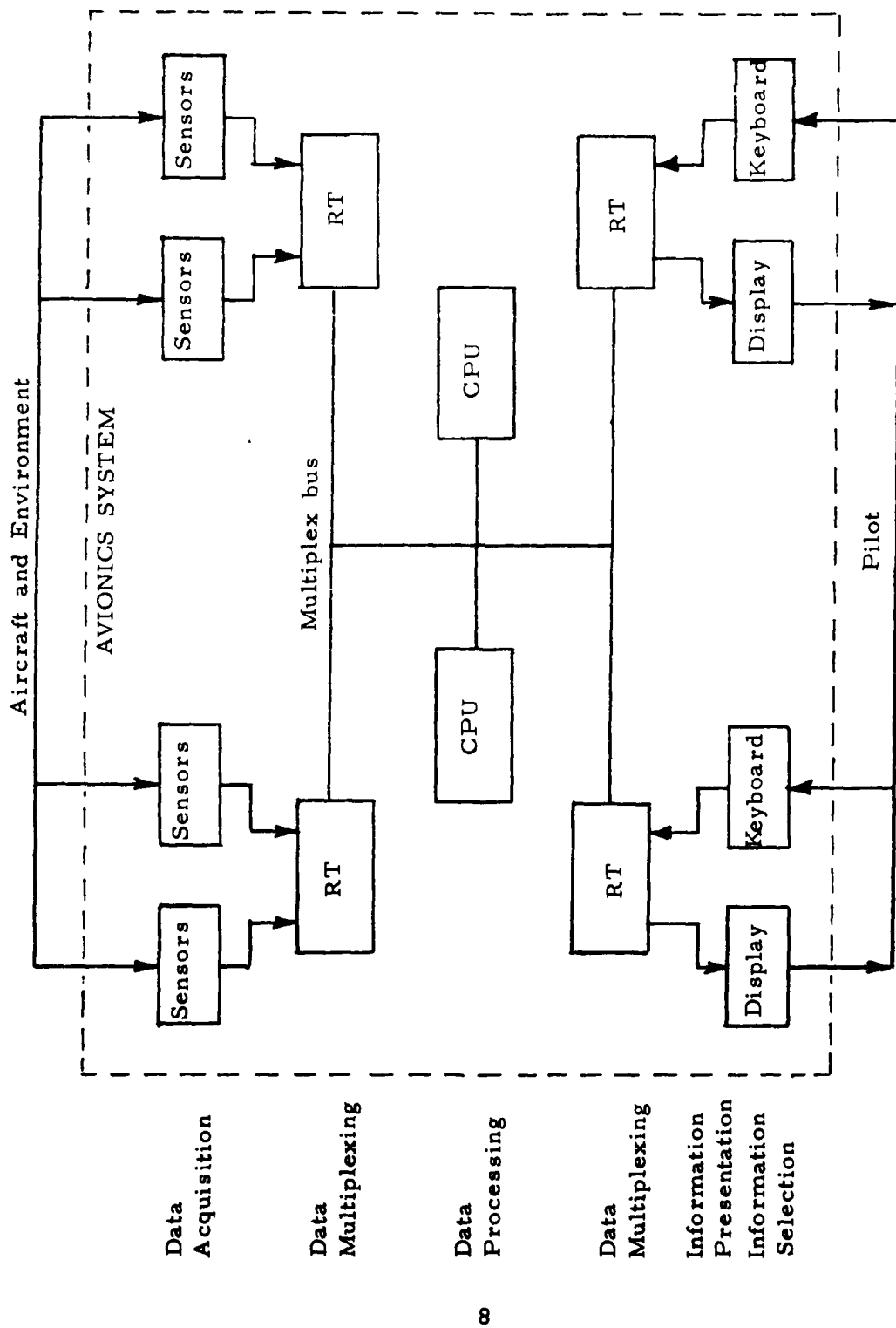


Figure 2 Architecture of an Integrated Digital Avionics System

remote terminal unit (RTU) which performs signal conversion, concentrates incoming signals, and distributes outgoing signals.

RTU's are hardware devices developed by hardware personnel and were originally introduced into avionic systems for the express purpose of reducing the weight and space required by the cables between subsystems and the system CPU. The RTU provides for a reduction of cabling by multiplexing data originating from sensors and switches onto a common bus and by demultiplexing data from the common bus which is destined for displays. In reducing the problems of weight and space created by cables between remote subsystems and system CPU, the RTU effectively transfers the hardware interface that logically exists between subsystem cables and CPU from the CPU to a location intermediate to subsystems and CPU. In the remainder of this section we will attempt to show that this dislocation of the subsystem/CPU interface is a major source of problems for maintenance and system design personnel and we will introduce the concept of the remote link unit (RLU) as a remedy for this situation.

The utilization of CPU's and RTU's in integrated avionic systems (Figure 2) is significant for the following reasons:

1. Overall hardware architecture and software organization is specified by a system designer. This person must take into account two distinct items:

- A. The commonality of data among functions, and
  - B. The interactions among the system's concurrent tasks.
2. In order to accommodate the variety of required signal interfaces several different types of standard interface modules are specified for use in the RTU's.
  3. The standardization of interface modules takes into account signal types and a protocol for controlling the transfer of data between a subsystem and the CPU through the interface module. Note that the interfaces are specified in terms of electrical signal characteristics and are not concerned with the information content of the signal.
  4. The standardization of interface signals in many cases induces a subsystem designer to introduce a microprocessor into the subsystem to satisfy the interface standards.

The existing structure of integrated avionic systems causes a serious communication gap between subsystem designers and system programmers. The introduction of the RTU has created a tendency to separate subsystem and system personnel and at the same time forces subsystem designers to have a detailed knowledge of overall system software and forces system programmers to have detailed knowledge of not

one but many avionic subsystems. Thus the inherent design of integrated avionic systems requires a maintenance technician, for example, to have a complete understanding of overall system diagnostics. This is unreasonable and in our opinion contributes to a loss of Air Force job efficiency and leads to major complications at the time of system integration.

The following subsection describes how each personnel-type views an integrated avionics system and offers a reason as to why the systems have been designed in the existing manner.

## 2.1 SYSTEM PERSPECTIVES OF DIFFERENT PERSONNEL

An autonomous subsystem avionic system has an architecture that provides an intuitive association of each function with a corresponding hardware unit. Thus, the system designer, subsystem designers, pilot and maintenance technicians each view the system similarly. Figure 3 illustrates the different views of the system by distinct personnel. As a result of the uniformity in system functions and implementation, it is not difficult for a maintenance technician to translate a pilot's malfunction report into a specific line replaceable unit (LRU) which can be repaired or replaced.

The design of an integrated avionics system should translate the system functional requirements into three

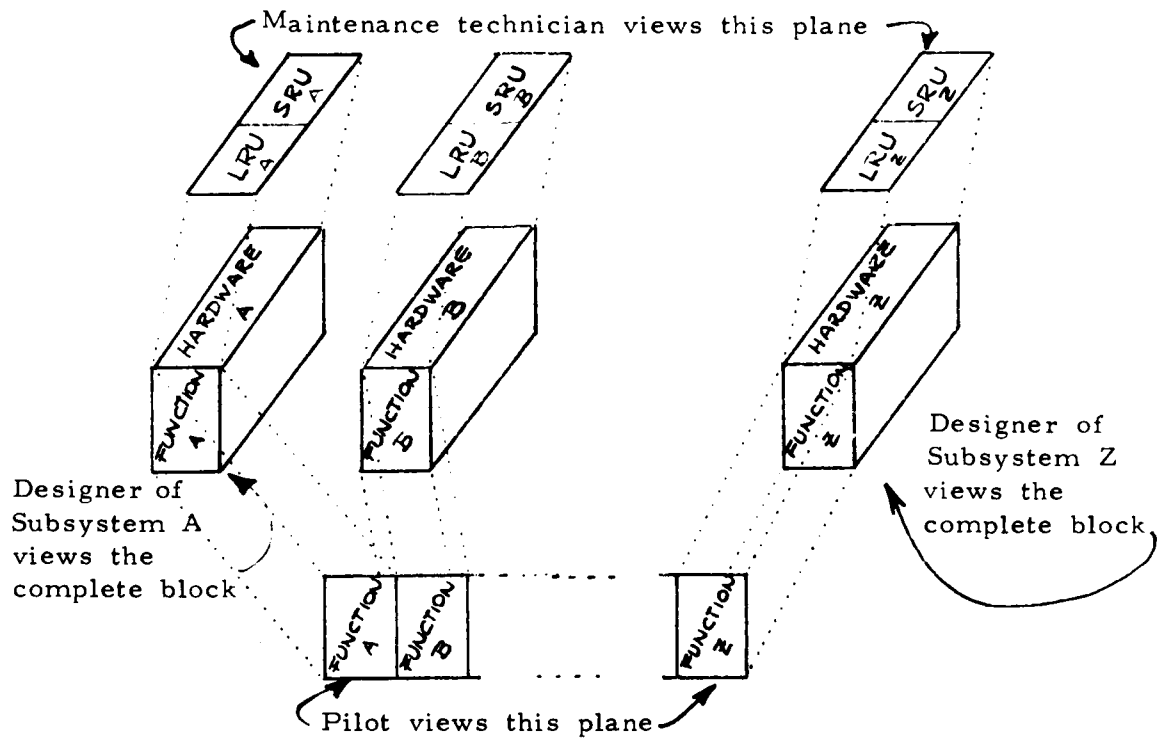


Figure 3 Conceptual View of an Avionics System Consisting of Autonomous Subsystems

distinct aspects: the selection of a hardware architecture, the definition of a software structure, and the specification of its internal operation. Figure 4 is a composite illustration showing the separate components such as software, hardware, LRU, functions, etc. of an integrated avionic system enmeshed on a block, and the same components in an exploded view. Each piece is labeled with particular system component(s), and the personnel who view that piece of the system (and therefore view or perceive the entire system as consisting of only those particular components). The only individual who must view the system in its totality as an integration of hardware and software functions, is the system designer. It can be seen that as a result of integration a technician must have system understanding similar to that of the system designer in order to be effective in performing maintenance. The variety of concepts utilized in the system implementation is beyond the technical training of most maintenance personnel.

As described previously the major deficiency in integrated avionic system operation is due to the fact that hardware and software interfaces for peripheral devices (subsystems) are often in unrelated locations. The hardware interface to a subsystem is defined at a connector which is clearly visible and accessible. The software interface to the same unit is embedded in the CPU operating system and

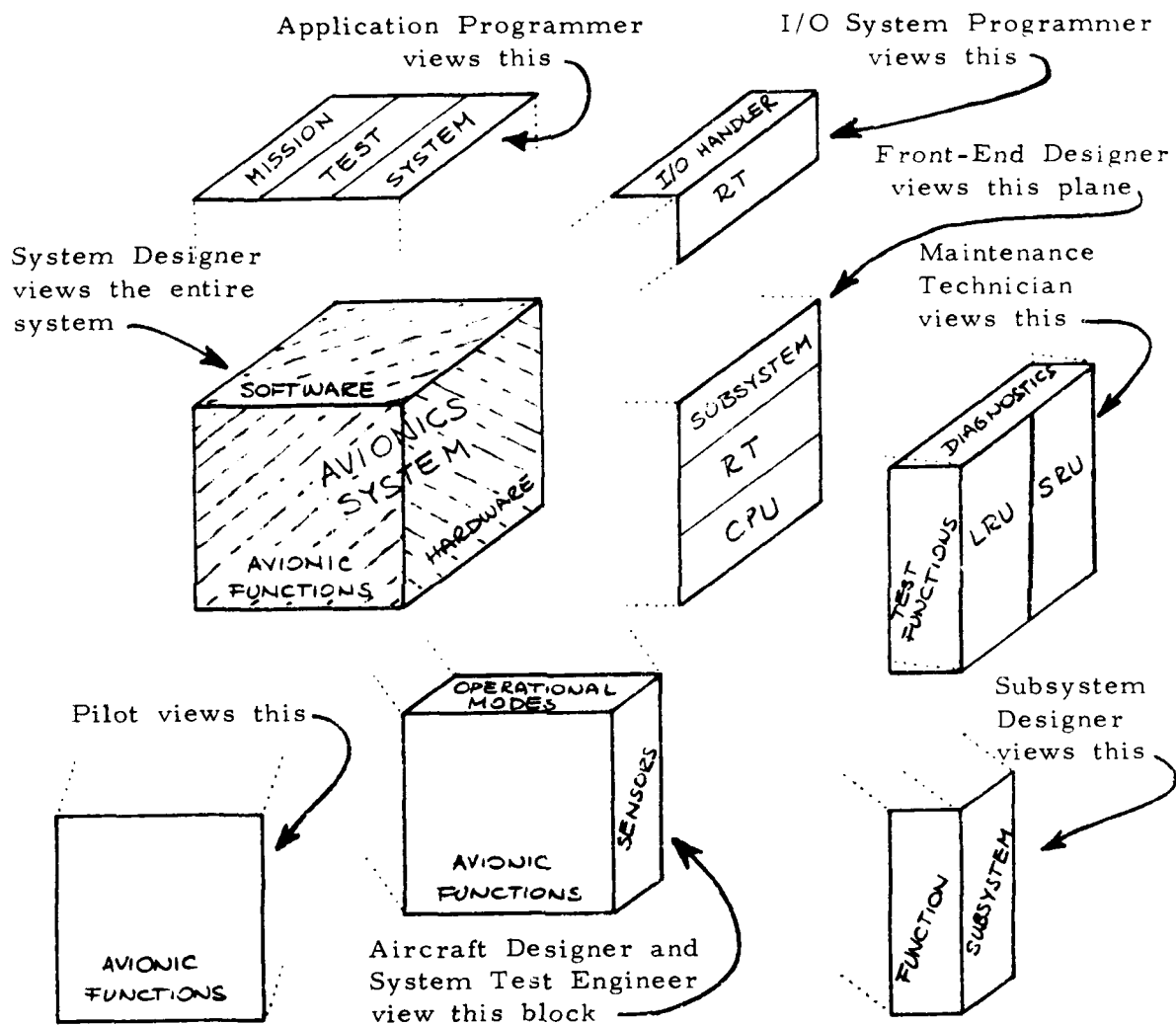


Figure 4 Conceptual Views of an Integrated Digital Avionics System by Distinct Personnel

usually is not immediately accessible or visible except to system programmers.

Consider a situation where the pilot observes an error in the aircraft navigational information. He will identify the variable which he considers unreliable. In establishing the cause of the problem, a maintenance technician must analyze the flow of information (data path) throughout the system. The raw data from one or more subsystems must be followed through one or more remote terminals into a global common area in the CPU's memory. This data is manipulated by device-dependent programs that convert it into a form that is compatible with the navigational data program. The processed data will be stored in a global common area, and then it will be routed through a remote terminal to a display. Notice that the error in navigational information could have been caused by any one of several hardware or software system components through which the signals traveled. An error could have occurred in the subsystem acquiring the data, or on the corresponding interface module of the remote terminal, or within the data conversion programs, or due to illegal interaction in the common area by other system programs, or at the display. To properly establish the source of the error, the maintenance technician must have a thorough understanding of how the total system operates in order to isolate the cause of failure. This

illustrates the level of complexity which is being confronted by technicians who maintain integrated avionic systems.

The root of this complexity resides in the prevalent design philosophy which emphasizes operational performance rather than ease-of-maintenance and fault tolerance. Fault tolerance and maintainability cannot be tacked onto a system to supplement the design of specified avionics functions as an afterthought. They must be an integral part of the architectural design and system specifications. By creating general purpose processing modules capable of supporting automated maintenance, and by providing these modules with a complete interface (hardware and software) it is possible to design avionic systems which are reliable, easy to maintain, and simple to understand. This modular design approach will simplify the work of hardware, software, and maintenance personnel and will lead to a precise standardization of interfaces and easy documentation of system operation.

In order to understand why integrated avionic systems have the existing architecture, one must review the evolution of supervisory control systems presently used in industry. Such review will reveal that the majority of industrial digital data acquisition and control systems have a similar architectural organization to that of present integrated avionic systems. Integrated digital avionic systems have inherited the architectural configuration of their

forerunner, the digital supervisory control system. However, the basic criteria for the design of industrial process control systems and for that of avionic systems are not the same. In fact, they have completely different driving requirements insofar as system design objectives are concerned. A supervisory control system is designed around a general purpose computer equipped with remote terminals which supports a variety of process control interfaces. The design objective is to maximize system adaptability to diverse processes at minimum cost. This design philosophy leads to the use of remote terminals and interface modules that provide greatest flexibility in adaptation to a variety of processes at the lowest possible cost. It also leads to centralized processing at the CPU and requires that system maintenance be performed by highly trained engineers with a thorough knowledge of the system. In industry this requirement does not create a problem since manufacturers of supervisory control systems also install and maintain their own equipment. As a result, customer service engineers who are highly trained in the specific processor, software and interfaces, can quickly isolate failures and repair the system.

Digital integrated avionic systems, on the other hand, have design constraints which are quite different from those encountered in process control [2]. Size, weight, and

performance are considerations of far greater importance than cost and flexibility of adaptation to distinct processes. Avionic systems are one-of-a-kind designs, with a very well-defined set of functions to be performed and are produced in large quantities for a specific aircraft. Such systems integrated with data acquisition functions and display peripherals constitute an extremely sophisticated, high density, electronic package with a complex system of signal cables. Access to the present equipment which is distributed throughout the aircraft is limited and constitutes a major hindrance to maintenance. In view of this, we conclude that the design objectives for digital avionic systems and for supervisory control systems are different and that avionic systems require automated maintenance support to compensate for the difficulty of system access and the limited knowledge of maintenance technicians in digital systems [2].

The RLU constitutes a first step in the direction of unifying hardware and software interfaces at one location. The RLU is the result of applying this design approach to replace remote terminals. The same RLU concepts can be applied at a system level to establish a framework for avionic system design.

## 2.2 THE COMPLETE INTERFACE

The concept of a remote link unit leads to the definition of a complete interface. Use of this interface in avionic systems will simplify their design, testing, quality assurance, integration, and maintenance.

A complete interface is characterized by total access to control, status and information of the interfaced device. The essential features of a complete interface are listed in Table 1. They include more than mere specifications of electrical connections, shared storage or status indication. The complete interface is a combination of all these functions with the assurance that it provides complete control, monitoring and testing of the internal elements of the units with which it interfaces.

The operator console of a conventional computer is an example of a complete interface. Through it, the operator may control the CPU and monitor the status of all programs being executed. By loading diagnostic programs from mass storage, he may establish the operational status of all major system components. Ideally, a complete interface should provide similar resources. That is, it should provide information, control and status of the device being interfaced. The interface between a link module and the link manager in a remote link unit is an example of such an interface. A link module provides control, status, and Table 1

TABLE 1  
FEATURES OF A COMPLETE INTERFACE

SIGNALS AND CONFIGURATION

- Standard Signals (function, level, and timing)
- Standard Configuration (control, status, and data)
- Standard Protocol (exchange of control, status, and data).

CONTROL AND STATUS

- Complete Controllability and Observability
  - Hardware Modules
  - Software Modules
- Operational Mode and State
  - Primary Functions
  - Testing and Fault Isolation
- Communication
  - Data Transfer Status
  - Error Detection and Recovery

INFORMATION

- Subsystem Identification
  - Type and Model
  - Function
  - Location
- Subsystem Data
  - Standard Format
  - Accuracy
  - Reliability

information on itself and on the subsystem to which it is connected. A subsystem interfaced through a link module operates as a single module with a complete interface at the link module. The subsystem data presented at the interface is free of device-dependent codes or formats. The value of acquired data is presented in engineering units together with a tolerance corresponding to a standardized interval of confidence.

To do this, the interface must be supported by a processor to perform data conversions, verify data validity, perform tests on the interface and subsystem, and maintain the standard format for complete interface configuration. The complete interface defines a natural boundary between system design and subsystem design. Figure 5 illustrates this boundary for a subsystem that provides aircraft altitude. The mission software is designed to accept the information of altitude in meters with a certain tolerance range. The altimeter provides this information through its complete interface, and therefore, the specific type of altimeter used and the manner in which it is interfaced is totally transparent to the system programmer. A maintenance technician may access this interface through the maintenance port of the remote link unit and make direct readings of altitude or run diagnostics in both the link module and the altimeter to determine operational status of each unit. The

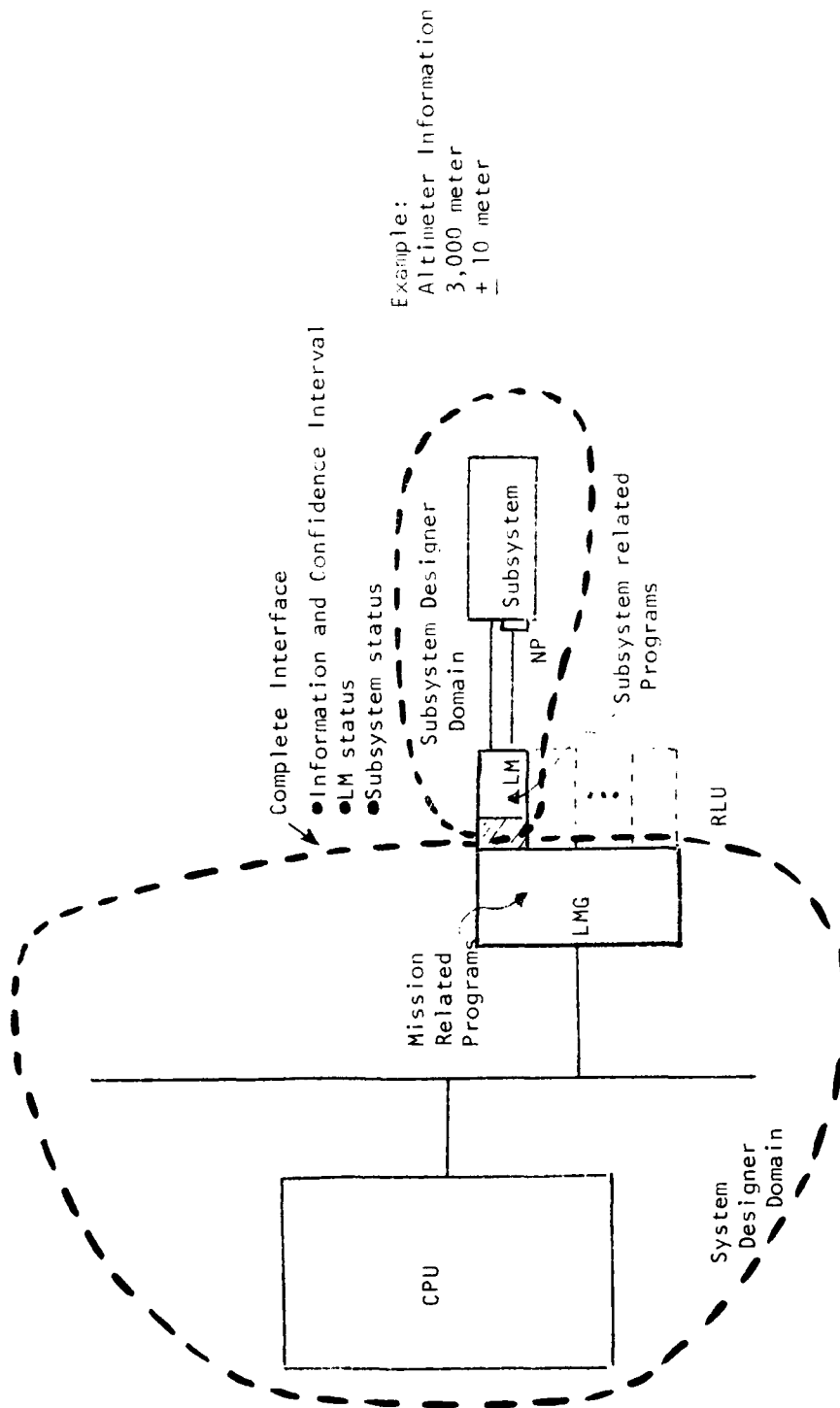


Figure 5 The Complete Interface as a Delimiter of System and Subsystem Design

interface then, by providing data on the operation of the link module, and the operational status of the altimeter, provides the maintenance technician with exact information on components that need replacement or repair.

The link module and its complete interface, should be made an integral part of the subsystem being designed. Thus, programs written for debugging the subsystem at the manufacturing plant will run on the link module and will not only be used at the factory, but also by maintenance technicians to determine causes of malfunctions. Likewise, the link module can run quality assurance final inspection tests with a go/no-go result. These programs may be used in normal operation to determine the conditions of the link module and the subsystem during a mission. This approach of providing the link module as a processor used as an extension of the subsystem and on which all testing and quality assurance programs will reside, will provide the Air Force with high quality programs and will eliminate duplication of efforts of having distinct set-ups for factory check-out and for maintenance crews to utilize. Another important aspect is that the link module will constitute a vehicle for maximum transfer of information from subsystem designers to maintenance technicians.

### SECTION III

#### DESIGN FOR MAINTENANCE AND REPAIR

Design for maintenance must take into account all aspects required for successful maintenance and repair. These include failure detection, failure reporting, failure isolation, repair verification and calibration. The techniques used to implement these capabilities should be compatible with both partial and total automation. The ultimate objective of design for maintenance is achieving maintenance and repair without human intervention.

The design-for-repair techniques which may be supported by the RLU are presented in this chapter in the manner in which automated maintenance would take place. The first topic considered is fault monitoring and recording. This is followed by a presentation of techniques which may be used to support repair. The final section presents techniques of automatic calibration which may improve the operational performance of the system and provide a basis for incorporating preventive maintenance. A comprehensive list of publications on maintenance oriented design is presented in reference [3].

### 3.1 FAILURE DETECTION AND REPORTING

RLU based systems are capable of performing several on-line functions aside from the expected functions of signal conversion and standardization. These tasks include the on-line detection and recording of subsystem faults, the execution of on-line diagnostics, and the recording of environmental conditions pertinent to the operation of the interfaced subsystems.

#### 3.1.1 Detection and Recording of Subsystem Faults

In RLU-based systems each subsystem (or portion of a subsystem) will be interfaced through a dedicated Link Module to the RLU Link Manager and the system data bus as illustrated in Figure 6. The Link Module has access to subsystem information in the associated subsystem electronic nameplate. This information includes the output signal levels expected from the subsystem, the timing of information expected from the subsystem, and the various diagnostic routines developed by the subsystem designer. This information is used by the Link Module to test subsystem responses according to their expected values, and to flag responses which are not as expected. For example, if a thermocouple develops an open circuit the signal from it will be out of range. The Link Module detects this condition and writes it back to the subsystem nameplate. Additionally, the

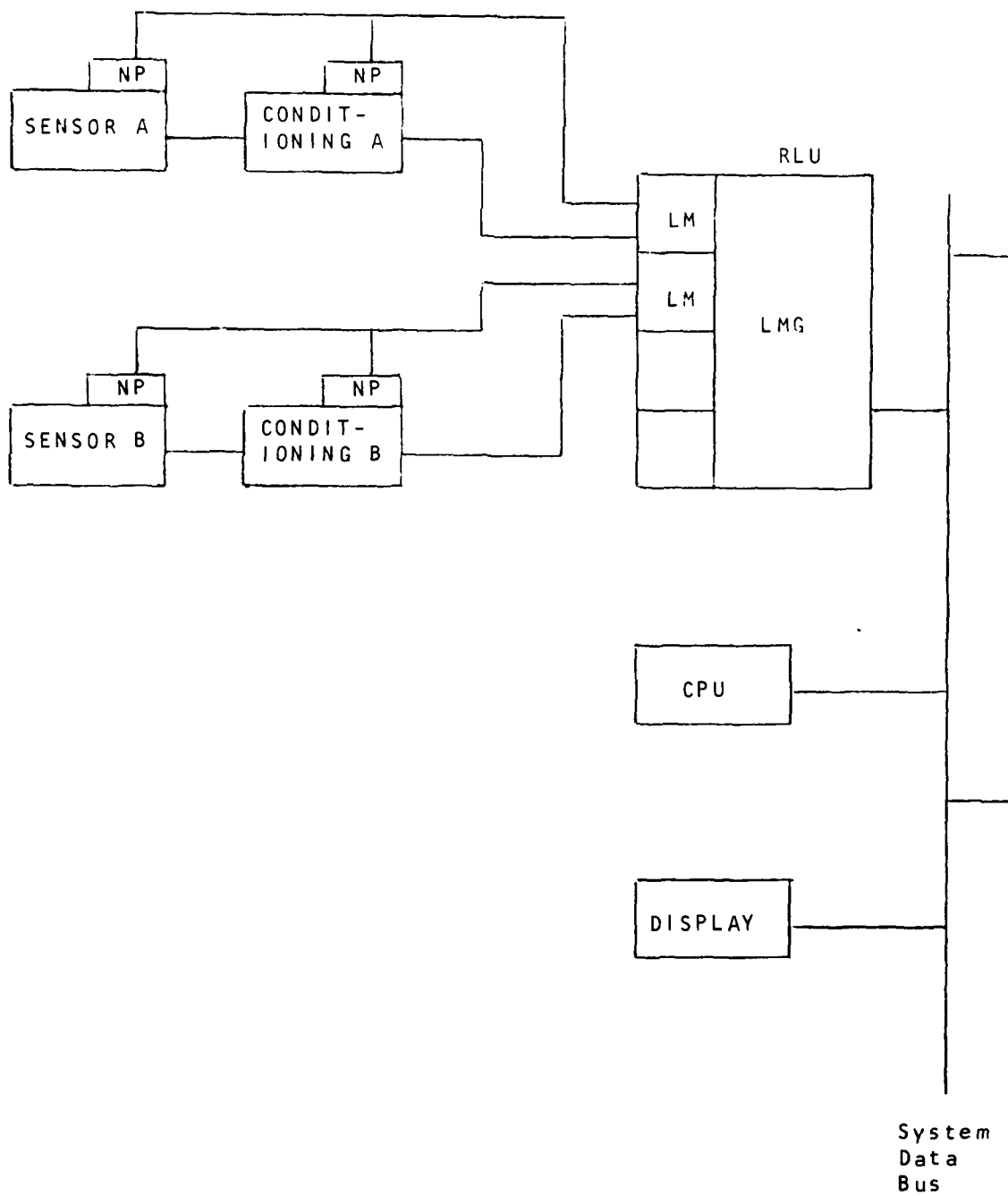


Figure 6 RLU-Based Avionics System

time of occurrence of the fault as well as other environmental information such as temperature may be recorded. This information will be useful to repair crews at a later time.

An example of a fault reporting system which involves the flightcrew members and utilizes the features of RLU's is the following: A "Maintenance Notebook" function could be provided by proper software in a system CPU. The notebook function and data entry could be implemented in the form of selection trees displayed on a cockpit CRT. The type of CRT display currently being developed for the DAIS system would be appropriate for this application as it contains selection keys whose function can be changed by using different displays on the screen. Once called by the flightcrew, the notebook function would ascertain which system was being reported and could display a menu of most likely system/subsystem failure modes for the flightcrew to flag. The subsystem nameplates will contain diagnostic programs and data provided by the subsystem designer. This data could contain a series of most likely failures and their failure symptoms which could be used by the notebook program. By simple keyboard responses the flightcrew could record noted discrepancies, and the information stored in the corresponding RLU for later use by maintenance personnel.

### 3.1.2 On-Line Diagnostics

Subsystem diagnostic routines which can be performed on-line should be provided by the subsystem designer. The thermocouple example of subsection 3.1.1 described tests which might be performed during normal on-line operation of the subsystem. Thus, the RLU can support fairly sophisticated on-line subsystem evaluations. These could range from simple limit checks to subsystem emulation performed by Link Modules dedicated to this task. For subsystems designed to interface with an RLU the Link Module will have access to key test points internal to the subsystem. By utilizing proper subsystem design, comprehensive self-tests may be performed.

Figure 7 illustrates a method which could be used to test major portions of a simple analog data channel. The test consists of changing the gain of the signal processor by a known amount through the use of a Link Module controlled switch. By measuring the signal before and after the known gain change is introduced, the Link Module can ascertain whether the result is as expected. This form of testing is attractive since with a small amount of additional hardware a subsystem can be tested end-to-end and in a manner which does not degrade the ability of the subsystem to provide information to the system.

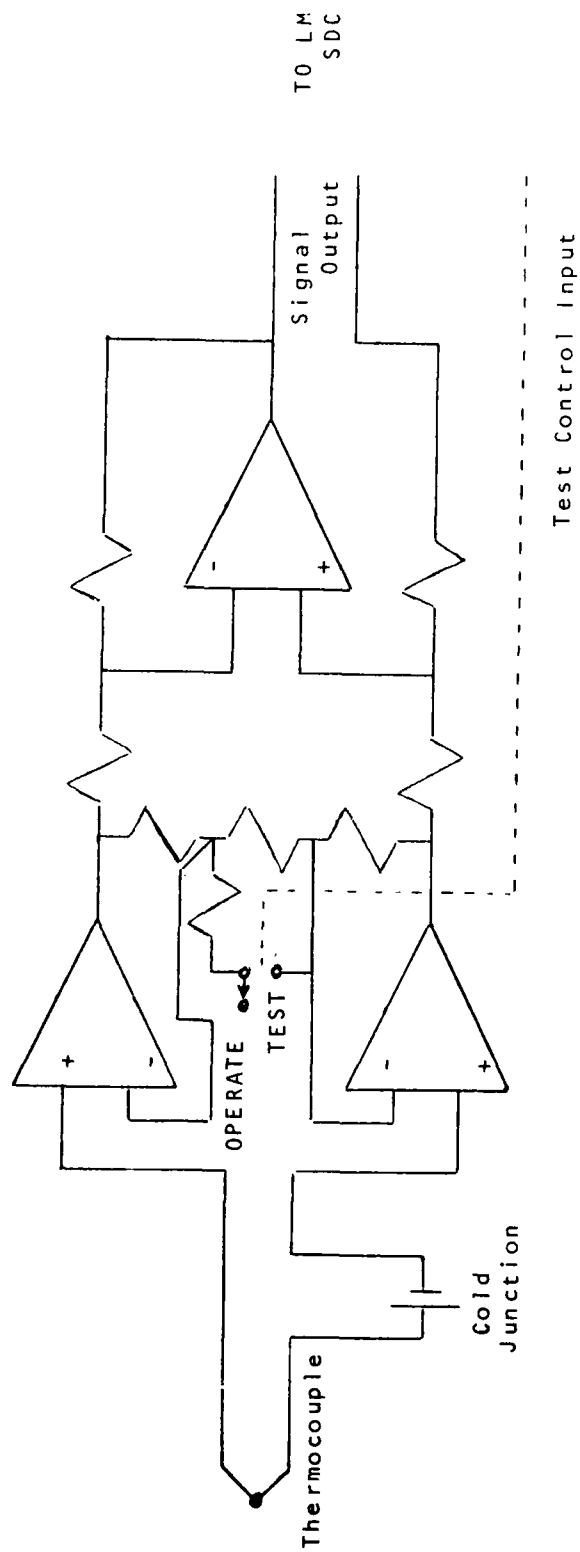


Figure 7 LM Controllable Test Example

### 3.1.3 Special Purpose Data Recording

The programs available to the Link Module allow subsystem designers to record information which is of interest to them and/or to repair crews. For example, the designer might need to know the total time his subsystem has been powered up and the environmental stresses to which it has been exposed. Operating time and temperature records can be maintained in the subsystem nameplate and periodically made available to the subsystem designer. This information would be of use to repair and maintenance personnel for the scheduling of periodic calibrations or for ascertaining whether observed faults may have been induced by the environment.

### 3.2 FAILURE ISOLATION AND REPAIR

The RLU should play a central role in the repair and maintenance of avionic systems. It will influence the required level of Aerospace Ground Equipment (AGE), the number of personnel required, the generation and distribution of repair logs, and the actual repair process itself.

Current repair procedures often require special purpose test equipment to adequately evaluate a given subsystem. The use of an RLU in the design and implementation of avionic systems should result in a reduction of the required specialized test gear. This will result from the

standardization of signal interfaces as well as the availability, within the LM's Interface Configuration Adapter [1] (ICA), of a general purpose piece of test gear associated with each interfaced subsystem. The ICA/Link Module has the ability to generate test signals as well as measure and analyze subsystem responses.

If the Link Module/Subsystem interface is properly designed, the routines provided within the subsystem by the subsystem designer should be able to locate failed subsystem components with a high degree of reliability. Recall that these routines will be developed by the subsystem designer who should have an accurate and comprehensive understanding of subsystem operation and who therefore will be in the best position to design meaningful diagnostic tests and fault interpretation routines. We believe that digital signature analysis techniques [4] might be useful to increase the resolution of the fault isolation capability of an RLU based test program. This should allow for the field repair of subsystem faults which previously required the assistance of a higher level repair activity. Subsystems which have been designed to use standard electronic components will be especially amenable to the signature analysis approach.

The subsystem electronic nameplate may be used to store the results of on-line and off-line diagnostics as well as the subsystem anomalies recorded during subsystem activation.

This storage medium, uniquely designed for each subsystem, should reduce the amount of hardcopy records which are required to support the maintenance operations. Repair logs may be maintained for subsystems and be immediately available to the repair crew. The data on subsystem failures contained on the nameplates may supplement, if not entirely replace, the crew debriefing information which is often inadequate.

Several of the applications described above are based on the use of the LM's ICA. The ICA can perform a comprehensive self-test by taking advantage of the wrap-around nature of its signal interface. This is described in the report "Functional Design of the Remote Link Unit: An Advanced Remote Terminal for MIL-STD-1553B," [1] section 4.4.

The ICA can support testing of the cables used to connect the LM to a subsystem. The subsystem end of the interconnecting cable could be attached to a connector whose pin-to-pin configuration was known to the Link Module through information stored in the subsystem nameplate. The ICA could then be used to verify the pin-to-pin configuration of the test connector by exciting a wire in the cable and verifying that the appropriate corresponding return wire demonstrated continuity.

### 3.2.1 Repair Centered Reports

Figure 8 illustrates the configuration of an RLU based avionics system and indicates the three report levels which may be supported. This subsection will describe the three reports to be used to support the repair function and will also indicate how the data is generated, stored, and accessed.

Level 1 reports are available from the CPU. The data for this report is stored in the CPU memory and include the items listed in Table 2. System bus failures are noted by the system processor and a real time record of this type of failure is maintained within the system data base. Unexpected responses, parity failures, 'dead' RLU's and/or subsystems and an unusable channel are recorded and summarized to form a part of the Level 1 report. The results of CPU self tests (arithmetic unit tests, peripheral controller tests, memory tests, etc.) are similarly recorded and reported. The level 1 report includes items pertinent to the operation of the CPUs, the system bus, and the RLU/bus interface. In addition, the level 1 report includes observed failures at lower levels such as out of tolerance results from subsystems. This data allows error isolation by comparing level 1 reports with level 2 and 3 reports. In general, any anomalous data or behavior observed on the system bus is reported in the level 1 report.

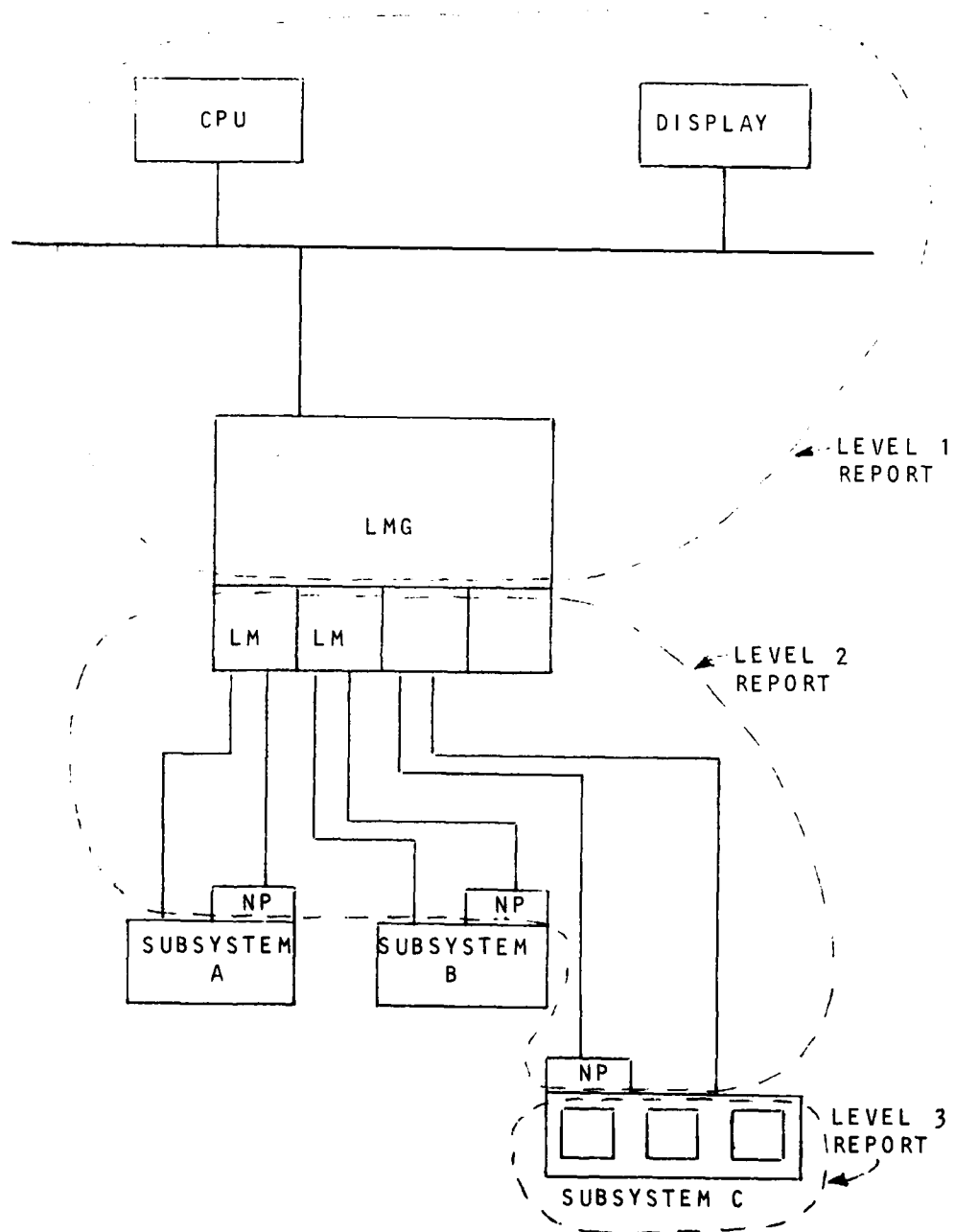


Figure 8 Reports Maintained by an RLU-Based Avionics System

TABLE 2  
LEVEL 1 REPORT DATA BASE

On-line Observations

Failures on the system bus  
    errors during bus transactions  
    non-responding RLU's  
    non-responding subsystems

CPU failures  
    arithmetic logic unit failures  
    peripheral controller failures  
    memory test failures  
    program errors

Out of tolerance results from lower levels

Off-line observations

Diagnostic program induced errors

Level 1 reports are accessed via appropriate software executed in a CPU using the normal cockpit display and entry devices. It is desirable to be able to generate this report from any processor as one of the master processors may be damaged. The bus presently provides a redundant path in the event of errors occurring on one of the channels.

The level 1 report can be of varying degrees of sophistication. The simplest format is a tabulation of system discrepancies noted during the last flight. This report could be valuable as an adjunct to the standard flight debriefing information available to the maintenance crew. A more sophisticated report-generating program would attempt to evaluate errors and to determine which system components are suspect. With the information available in a level 1 report, several classes of failures may be readily diagnosed. These include internal memory failures in the CPUs and failures on the system bus. A final enhancement may be a program which attempts to duplicate and isolate the failures in an automatic fashion. For example, if parity errors are found when two specific RLU's are communicating the system CPU may exercise the faulty link to pinpoint the origin of the failure.

Level 2 reports originate within the individual RLU's. The data for this report is stored in the RLU mass memory and contains the items indicated in Table 3. On-line failures

TABLE 3  
LEVEL 2 REPORT DATA BASE

On-line Observations

RLU failures

- LMG self test failures
- LM self test failures
- errors on the Subsystem Information Channels
- errors on the Subsystem Data Channels
- data bus errors

Subsystem failures

- out of tolerance results
- subsystem generated error status
- non responding subsystems

Off-line Observations

RLU diagnostic failures

Subsystem diagnostic failures

are detected and recorded during normal system operation by software executed in the RLU link manager.

The level 2 report may be accessed either via the maintenance port of the appropriate RLU or through the system bus. The report-generating software is part of the Link Manager software. The simplest report would consist of a table of observed anomalous results, arranged in the order of their occurrence. This simple report may be a useful diagnostic aid when used in conjunction with the standard debriefing forms. The data available in the level 2 (on-line) data base may also be used to automatically indicate suspected failed components and/or subsystems if adequate analysis software is available. Also, the RLU may be used to perform off-line diagnostics using software available in the RLU nameplate. The results of these off-line diagnostics forms the off-line portion of the RLU data base and again may be analyzed and used to pinpoint potential failures.

Level 3 reports use data stored in the subsystem nameplates as listed in Table 4. Parity errors on the Subsystem Data Channels (for the digital data paths) are noted by the subsystem link module and recorded in the appropriate subsystem nameplate through the nameplate interface controller. By using limits available to the link module from the subsystem nameplate, out of tolerance results

TABLE 4  
LEVEL 3 REPORT DATA BASE

On-line Observations

Subsystem failures

errors on the SDC

out of tolerance results

non-responding subsystems

Off-line Observations

Results of off-line diagnostics

are similarly noted and recorded. Parity errors on the Subsystem Information Channels are noted in the level 2 report as the SIC is not available for recording to the subsystem nameplate.

The level 3 report is available from the RLU maintenance port. The information is recovered from all subsystems associated with the RLU.

To summarize, the level 1,2 and 3 reports contain information on the CPU - system bus - RLU interface, the RLU - Subsystem interface, and the subsystem respectively. Considered as a group the reports can pinpoint system failures. Thus, if an error is reported for communications between RLU # 3 and RLU # 7 in the level 1 report it is not possible to determine where the failure occurred. The failure might have originated in either RLU or in the CPU which generated the level 1 report. Taking level 2 reports from each of the suspect RLU's in addition to using the level 1 report will clarify the location of the problem. If neither RLU has observed the errors there is a significant probability that the problem is with the reporting CPU. These kinds of considerations indicate that an overall report generating program, which would execute off-line in a system CPU, and which would have access to the data bases of the level 1, 2, and 3 reports, could pinpoint failures in many cases. In principle this approach could be extended into a

"superdiagnostic" with access to the subsystem diagnostics stored in the subsystem nameplates and could greatly facilitate the maintenance and repair of avionics systems.

### 3.2.2 RLU Based Repair Flowchart

Figure 9 illustrates the repair process expected for an RLU-based subsystem. There are two significant differences anticipated between current practice as outlined in the Design-for-Repair Concept Definition (DRCD) reports [5] and the maintenance practice for an RLU based system.

It is expected that the paperwork required to support the repair process in an RLU-based system will be significantly less than that required presently. This is due to the recording and documenting capabilities inherent in the electronic nameplates of RLU systems. A large amount of information such as calibration parameters, test procedures, equipment status, and repair diagnostic aids, will be stored in the various nameplates, obviating the need for maintenance personnel to gather this information except in rare instances (for example, if the nameplate itself has failed). In addition, much of the maintenance results will be stored in the nameplates, again eliminating the need for separate documentation. These results are accessible off-line. The failed LRU's will therefore be self-documenting as the nameplate will go with the unit to the next level of repair.

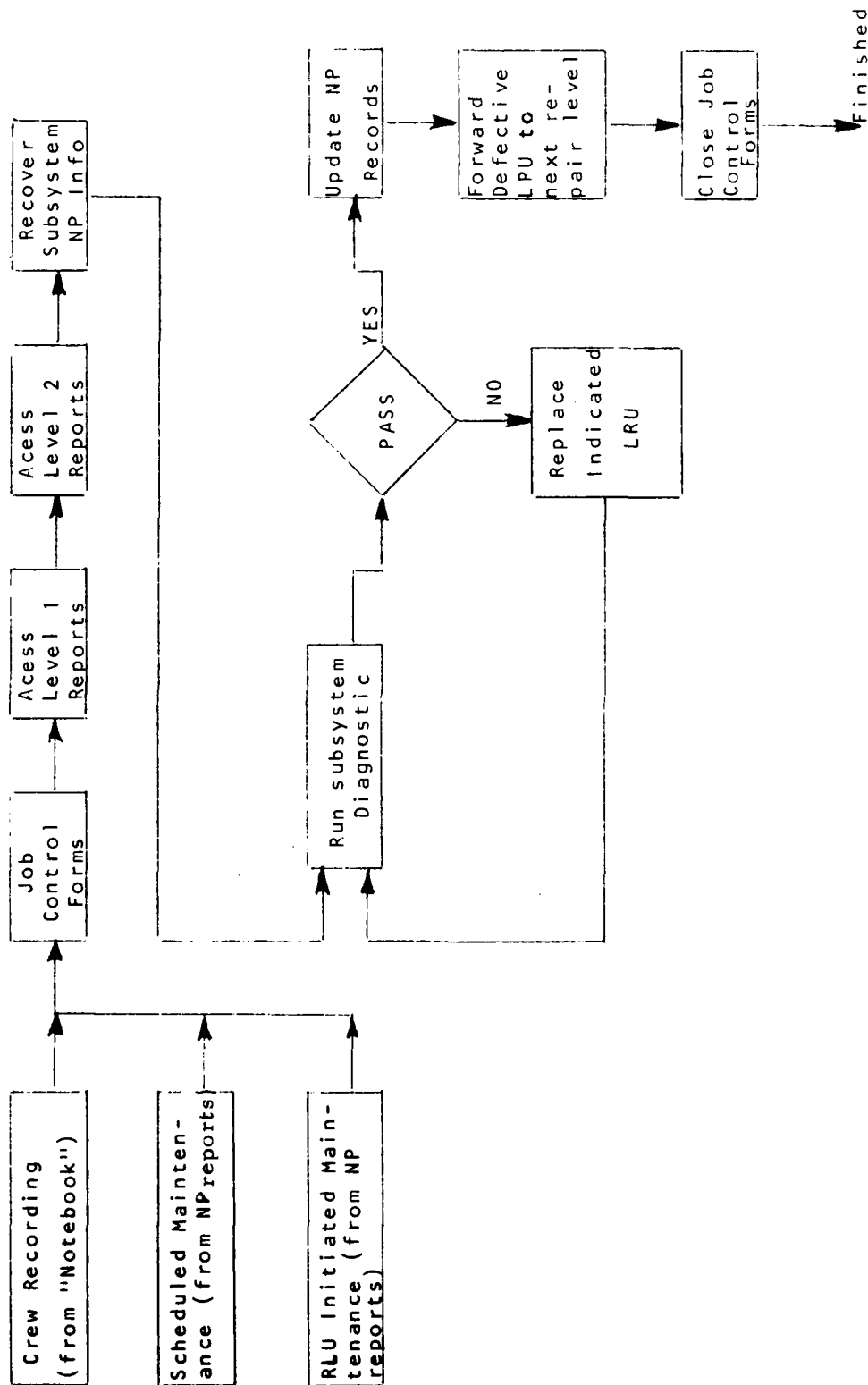


Figure 9 RLU Maintenance Flowchart Organizational Level Repair

This should improve the communication between organizational and intermediate repair operations and at the same time eliminate costly paperwork.

The RLU-based system should offer much faster repair times than an equivalent conventional system. This will result from the availability of sophisticated off-line diagnostics as described previously. It is expected that the use of such diagnostics will greatly improve the fault isolation skills of maintenance personnel and reduce the number of LRU replacements needed to repair a subsystem. The verification of the repair will also be assisted by the RLU.

### 3.3 REPAIR VERIFICATION AND CALIBRATION

Following repair the RLU may be valuable in verifying that the repair was effective in correcting the fault. Built-in diagnostics may be used to check out the operation of the subsystem in a simulated on-line mode.

The RLU based avionics system supports maintenance and repair at many levels. A global support concept is described in this subsection. As presently envisioned, the three report levels described in Section 3.2.1 may be combined in an automated operation to confirm overall system function for each aircraft, provide statistical reports on the equipment operation in several aircraft, prepare maintenance schedules, and pinpoint problem areas in need of attention. Figure 10

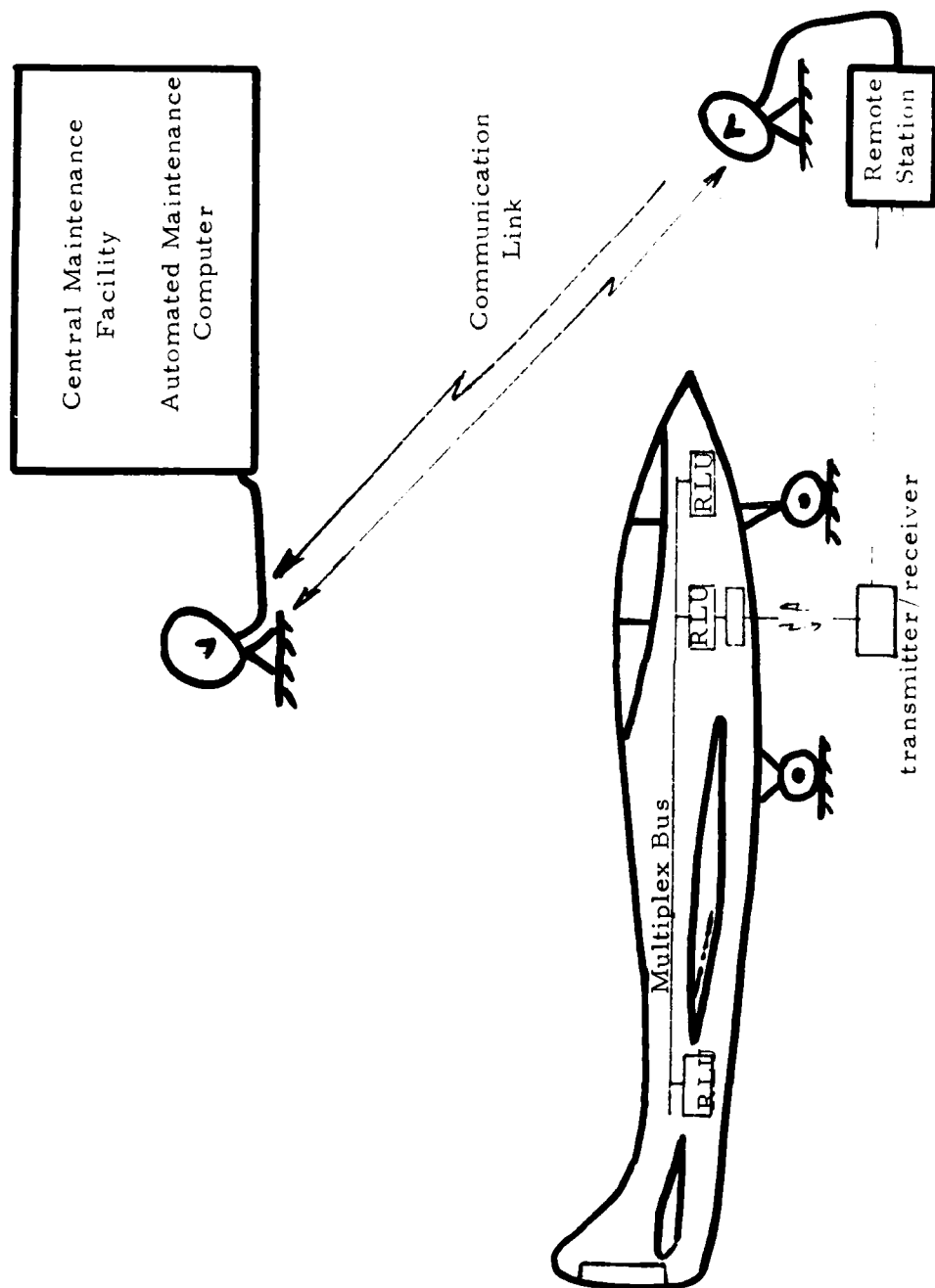


Figure 10 Routine Functional Confirmation Test Configuration

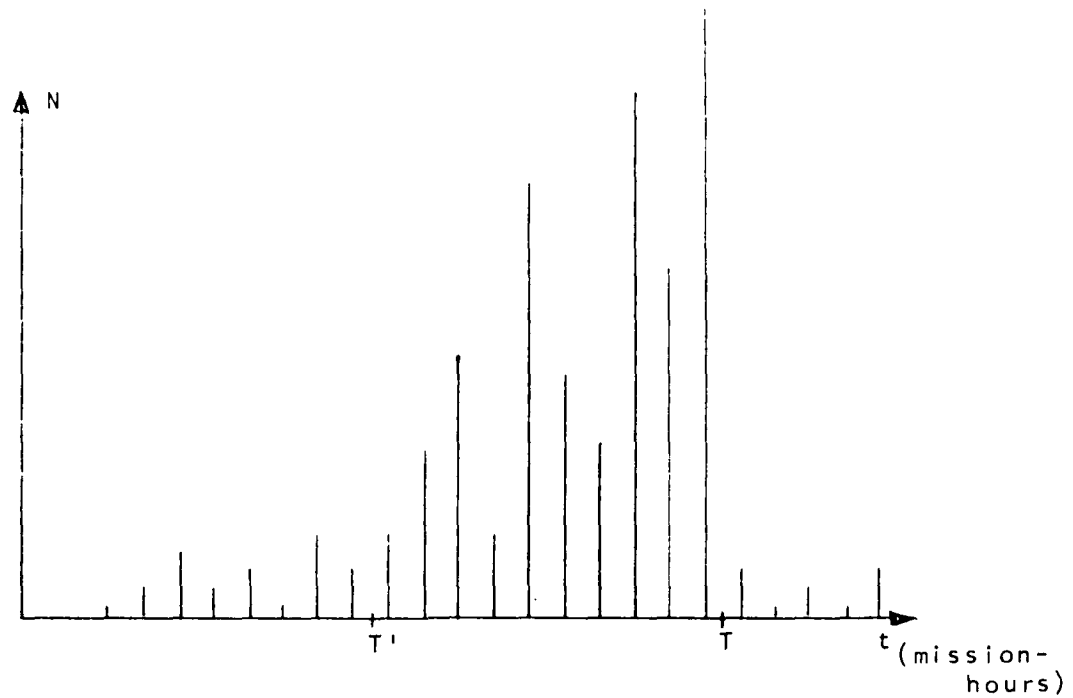
illustrates the concept. On a routine basis (for example, following each flight) the aircraft is connected to a central computer facility. The connection may be via a modem which is plugged into the system bus or over an RF link using existing aircraft radio equipment. From this connection the central computer has access to the three level report data bases previously discussed, as well as additional nameplate information such as diagnostics and scheduled maintenance intervals. By centralizing the computer function for a group of aircraft, meaningful statistics concerning the reliability of various avionic system components may be obtained. Maintenance schedules of both routine maintenance as well as maintenance indicated by observed failures may be generated by the central computer facility and made available to maintenance crews.

The central computer facility has access to the system bus and can therefore initiate and obtain the results of subsystem diagnostics. This operation would be useful in preparing schedules of required maintenance as well as confirming the operational status of the avionics system.

The calibration cycle is supported by the RLU. Each subsystem's nameplate can be used to store appropriate calibration data for the subsystem. In addition, since the nameplate information is used to provide offsets and scale factors for engineering unit conversions, the nameplate can

be an active component in the calibration cycle. For example a subsystem which provides an amplified thermocouple output might use the Link Module to convert the non-linear thermocouple output to a linear analog of temperature. The conversion equation coefficients, stored in the nameplate, might be updated during calibration to process out the long term drift in the thermocouple signal processing circuitry. This "software" calibration can be an alternative to subsystem adjustments.

The RLU recording system can also be used to correlate calibration cycles with subsystem performance. This comparison can be used to meaningfully extend or shorten the period of routine calibrations. Figure 11 illustrates an example which indicates that the higher failure rates associated with calibration period T can be avoided if a shorter calibration period such as T' were used.



$T$  = calibration period

$T'$  = reduced calibration period to  
reduce subsystem failure rate

$N$  = number of RLU reported and verified  
subsystem failures/100 mission hours

Figure 11 Calibration Cycle Adjustment Example

#### SECTION IV

#### DESIGN FOR FAULT TOLERANCE

Current methods of system design are based on fault avoidance and manual maintenance. The equipment is designed to be as reliable as possible and when failures occur, manual methods must be used to correct the fault. Typically systems designed under this philosophy are vulnerable to the failure of a single component.

The philosophy of fault tolerant design is somewhat different. For the purpose of this report we will define fault tolerance as the ability of a system to survive one or more physical failures of its components and continue to perform a meaningful percentage of its original function [7]. This design approach is an attempt to ensure system function in the event of single component failures.

Fault tolerant design is subject to its own set of tradeoffs. The initial equipment cost is usually higher due to the necessity of implementing redundant hardware. This is partially balanced by being able to utilize lower reliability (and hence, lower cost) components since a component failure is not catastrophic. The initial cost is also higher due to the more complicated design required by fault tolerant systems. The fault tolerant system provides a higher

availability and may have a lower life cycle cost since manual repairs can be scheduled at convenient intervals rather than immediately following a failure. This is due to the fact that the system continues to function even after a component failure. This flexibility in performing manual repairs allows the elimination or reduction of local repair operations and concentrates limited maintenance budgets in more cost effective centralized repair depots.

#### 4.1 SYSTEM DESIGN

Two approaches to fault tolerant system design are illustrated in Figure 12.

The first approach (architecture) is referred to as "Active Redundancy" [6] whereby the  $n$  channels are all active and a voting system is used to direct the output path to the channel which is most likely correct. For the approach illustrated (2 out of 3) a failure in a single channel would allow the voter to choose either of the two channels which agreed.

The second structure illustrated in Figure 12 is referred to as "Standby Redundancy" [6]. In this approach to fault tolerance a channel is used until it fails at which time a second channel is activated. This method requires a more sophisticated fault detection process than the Active Redundancy method.

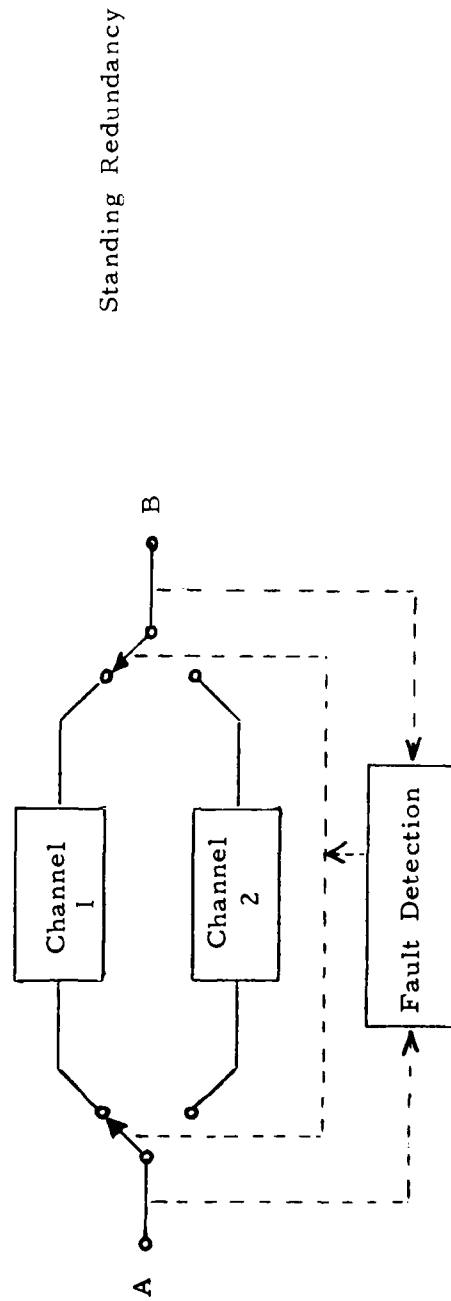
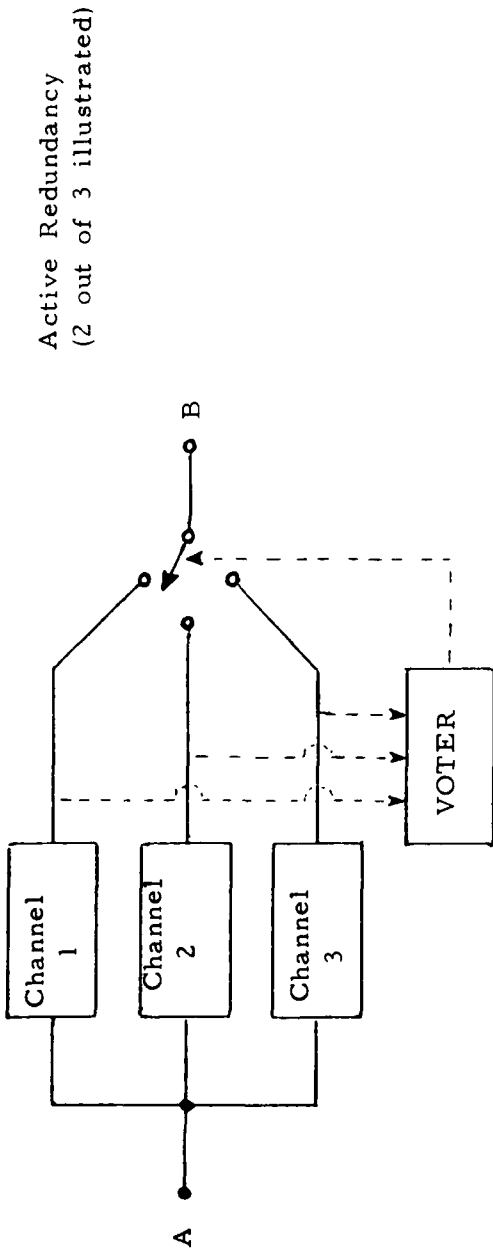


Figure 12 Two Approaches to Fault Tolerant Design

With the following assumptions, the reliability of the above methods can be compared to the reliability of a system which uses a single channel:

(a) The voter, fault detection circuitry, and switches are much more reliable than the channels being controlled.

(b) The channels all are equally reliable with a probability of failure of  $q = 1-p$ .

For the Active Redundancy ( $k$  out of  $n$ ) case, the system reliability is

$$R_{ar} = \sum_{i=k}^n \binom{n}{i} p^i (1-p)^{n-i}$$

which reduces to

$$R_{ar} = 1 - (1-p)^3 - 3p(1-p)^2$$

for the 2 out of 3 case.

For the Standby Redundancy case the system reliability is

$$R_{sr} = \sum_{m=0}^{n-1} P_m$$

where

$$P_m = \frac{p(-\ln p)^m}{m!}$$

For  $n = 2$  this reduces to

$$R_{sr} = p + p(-\ln p)$$

Figure 13 compares the reliability obtained for the two fault tolerant methods described above and a conventional system consisting of a single channel with reliability  $p$ . The standby redundancy technique performs best under the above two assumptions. It is of interest to note that active redundancy methods are inferior to conventional (non fault tolerant) designs for a range of  $p$  values [6]. Even so, the active redundancy techniques are often used due to their ease of implementation. The voter scheme used is the active redundancy method is a straightforward method of fault detection whereas the fault detection indicated for the standby redundancy method may not be easily realized.

In essence the standby redundancy approach requires an independent means of ascertaining the operational status of a channel.

#### 4.2 RLU USE IN FAULT TOLERANT SYSTEMS

The following five subsections describe in detail the methods by which the RLU can support fault tolerant subsystem design.

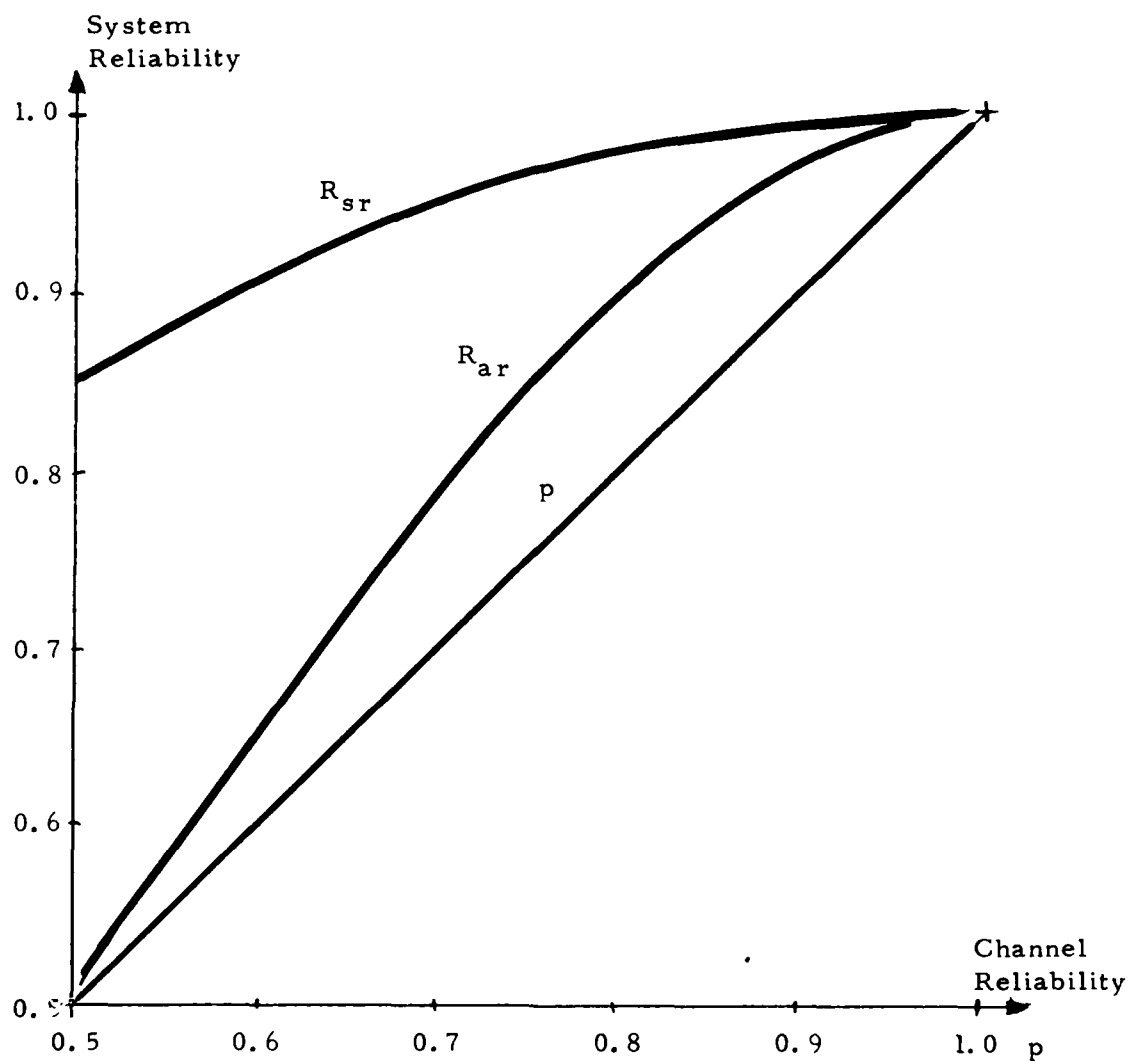


Figure 13 System Reliability for Single Channel ( $p$ ), Active Redundancy ( $R_{ar}$ ), and Standby Redundancy ( $R_{sr}$ ) Cases

#### 4.2.1 Active Redundancy

Figure 14 illustrates the use of an RLU as the voter in an Active Redundancy approach. A program executed in the Link Module provides the comparison and voting capability required. Although Figure 14 shows the use of a single LM, it is clear that numerous combinations of LM's and redundant subsystems are possible. When more than one LM is involved in a redundancy scheme overall coordination would come from the Link Manager.

The arrangement in Figure 14 is based on the assumption that the LM is more reliable than the interfaced subsystems. This is a reasonable assumption since the LM will be mass produced, is largely digital, and will likely be a product of very reliable VLSI manufacturing. The interfaced subsystems will likely not be produced by an equally reliable technology.

#### 4.2.2 Standby Redundancy

Figure 15 shows the use of an RLU in a standby redundancy situation. To take maximum advantage of this strategy the standby channel should not be powered until required [6]. This feature requires some means of controlling the power to the two channels as illustrated. The LM must be capable of determining the operational status of channel A. This may be done by software, or by sensing

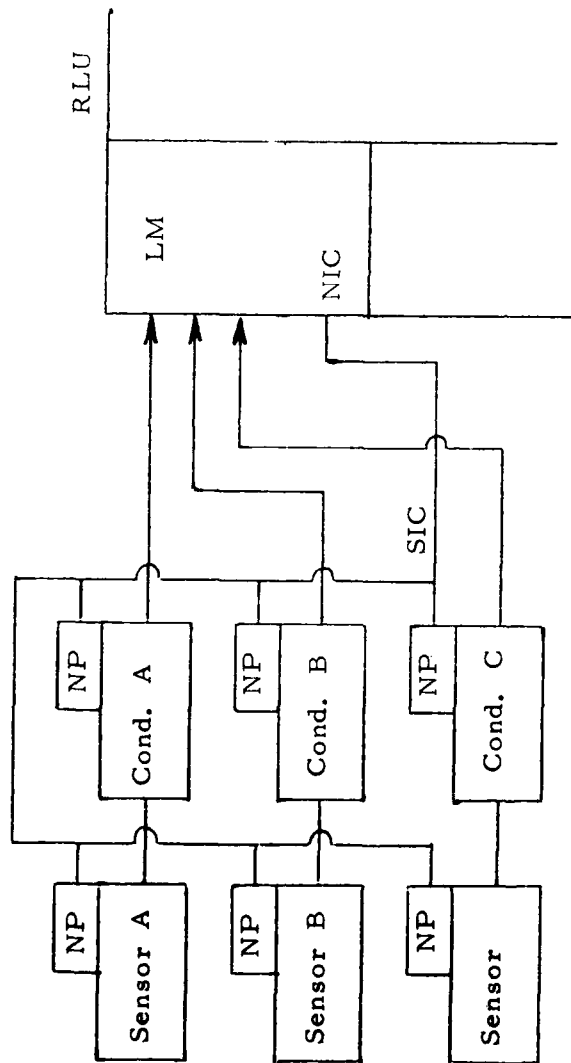


Figure 14 RLU Use in Active Redundancy Fault Tolerant Design

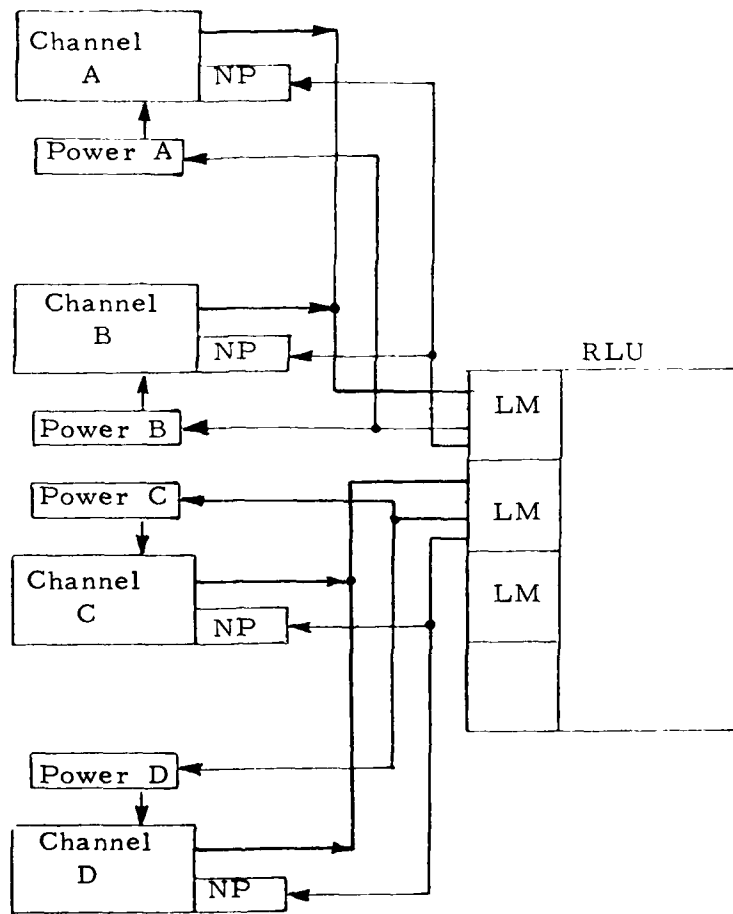


Figure 15 RLU Use in Standby Redundancy Fault Tolerant Design

BIT information from the interfaced subsystems, or by a combination of these techniques.

#### 4.2.3 Extension to System Level

Subsections 4.2.1 and 4.2.2 described means of incorporating redundant subsystem hardware and providing the RLU with the ability to select the hardware components to be used at any time. This concept can be carried to the next level in the system as illustrated in Figure 15. If subsystems A and B are redundant, then each RLU can select the most reliable subsystem as described above. In addition, the CPU can select the RLU/subsystem combination it believes is most reliable at any one time. If subsystems A and B perform different tasks, the CPU is able to access each subsystem through a choice of RLU, thus protecting the system function from failures in a single RLU.

Notice that many CPU/RLU/subsystem combinations can be configured. The combination which provides the highest reliability will be a function of the reliability of the various subsystems and will need to be determined in each specific case.

#### 4.2.4 Selective Redundancy

The approaches to fault tolerant design outlined above depend on an extensive amount of redundant hardware. The

capabilities of an RLU are such that alternate approaches to this "brute force" redundancy should be considered. By taking advantage of the distributed processing inherent in the RLU selective redundancy [2] can be utilized in key avionic subsystems and data paths which will result in a more cost effective approach to the fault tolerant system.

Figure 16 illustrates an example of selective redundancy in the LM/LRU interface. The LRU may be representative of a segment of analog signal processing in a subsystem. The amplifiers (A1 and A2) and multipliers (M1 and M2) combine to derive an accurate and easily used analog representation of a parameter of interest to the system. Normal operation would consist of the following:

1. keep the switch S1 in the position shown,
2. digitize the signal I1,
3. perform scaling operations (optional),
4. and present the result to the shared memory interface at the LMG along with a status word indicating normal operation.

Periodically the LM can use S1 to produce a sample of the less refined signal at I1. This signal can be modified by appropriate calculations in the LM to verify the operation of A1, A2, M1, M2 and associated components. It is apparent that there will be a time penalty associated with the digital processing discussed above as well as a loss of resolution.

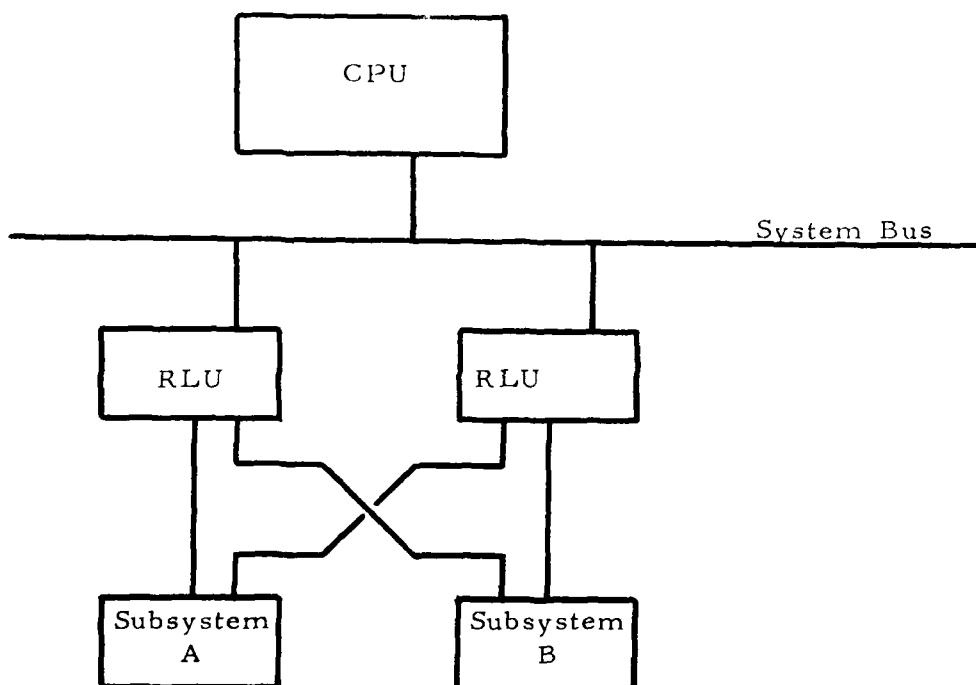


Figure 16 Redundant System Level Data Paths

This penalty may not be significant when the goal is to test the operation of the normal channel, an operation performed at a low duty cycle.

The interesting point of Figure 17 is that in the event of a component failure in the LRU (assuming S1 does not fail), the LM can still obtain useful data from the subsystem. It is likely that the variable being measured will not be known as precisely or as rapidly as before the failure, but a measurement can still be produced. The status word associated with the data may reflect the lower quality of the measurement.

A powerful extension of the above techniques can be envisioned. It may be feasible to make key measurements in different ways using different basic parameters. For example the aircraft position can be obtained from measurements by using known radio beacons or by the use of an inertial platform. The two determinations, using different sensors and different processing components, serve as backups to each other. The technique which results in the best measurement would normally be utilized however in the event of a failure the same information can be derived from measurements obtained independent of the failed subsystem. The distributed intelligence and decision making abilities of RLU's should make this technique useful.



Figure 17 Selective Redundancy at the LM/LRU Interface

#### 4.2.5 RLU Based Functional Clustering

Figure 18 illustrates an additional fault tolerant system architecture. This design allows the system to perform in a degraded mode in the event of system bus or CPU failure. If the RLU detects a CPU/RLU communication breakdown, the RLU has direct access to the cockpit displays. The pilot could select the alternate system operation suggested by the figure, i.e., allow the display to receive data directly from an RLU which had lost communication to the CPU. Presumably the display in this alternate mode of operation would be degraded but essential information could still be made available to the flight crew.

Figure 18 indicates a functional clustering of subsystems. This clustering is desirable so that, in the degraded mode, the RLU has access to all of the parameters relevant to a particular aircraft system such as navigation and can therefore process the parameters into meaningful display data. This approach is similar to the "computational subsystem" concept described in the Ultrasystems report [2].

#### 4.3 SUMMARY

This section has briefly reviewed several of the approaches to fault tolerant design which have proven successful in actual applications [7]. It is clear that the cost effective design of fault tolerant systems must be

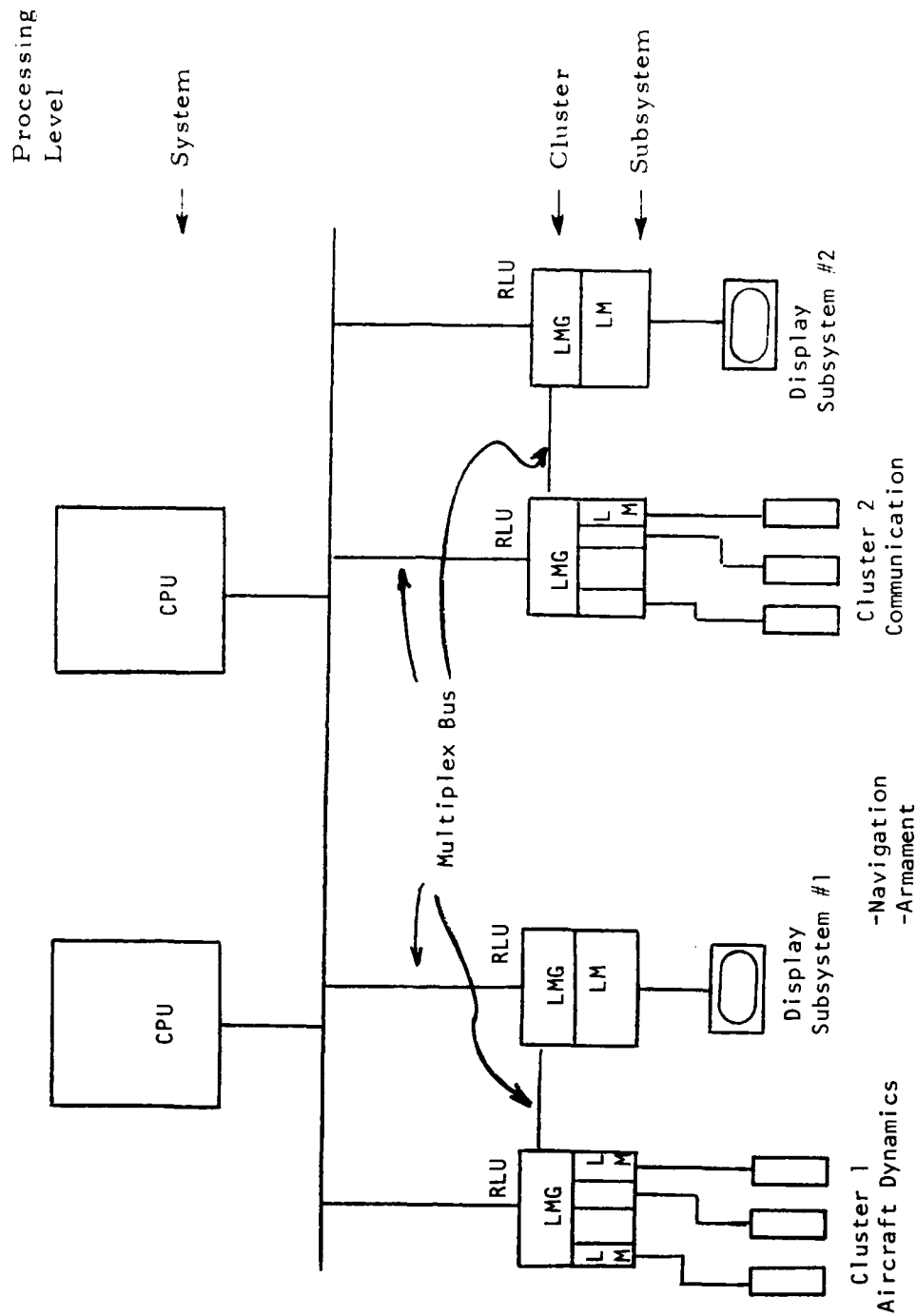


Figure 18 Distributed Processing with Functional Clustering

approached on a case by case basis. However, some generalizations can be made:

1. A distributed computational architecture [2], as results from the use of RLU's, is potentially more resistant to failures than a highly centralized one.
2. The blind application of redundancy will generally not result in an optimum system design. The use of redundancy is necessary for fault tolerance but it should be applied in those system components which are most critical to mission success and in a carefully thought out manner which utilizes the abilities of the RLU's.

## SECTION V

### MAINTENANCE PROGRAMS AND DATA STORAGE

The application of RLU concepts to design-for-repair and fault tolerance requires the development of special programs and the organization of data storage. These programs may be classified as maintenance-oriented or fault tolerance oriented. Each of these classifications can be further sub-categorized as system-oriented and subsystem-oriented. The separation between system-oriented and subsystem-oriented programs is established by the complete interface between link manager and link modules of an RLU. The proposed design for repair techniques requires a distributed network of data storage for support of automated maintenance. This data network has two distinct types of storage: mission-oriented failure storage and device-related electronic nameplate storage. The design of subsystem related programs should utilize a language that may be translated or compiled into a machine-independent (and therefore universal in nature) object code.

#### 5.1 SOFTWARE FOR MAINTENANCE AND FAULT TOLERANCE

The separation of system and subsystem softwares by the internal complete interface of an RLU yields a software organization that is depicted in Figure 19. System programs

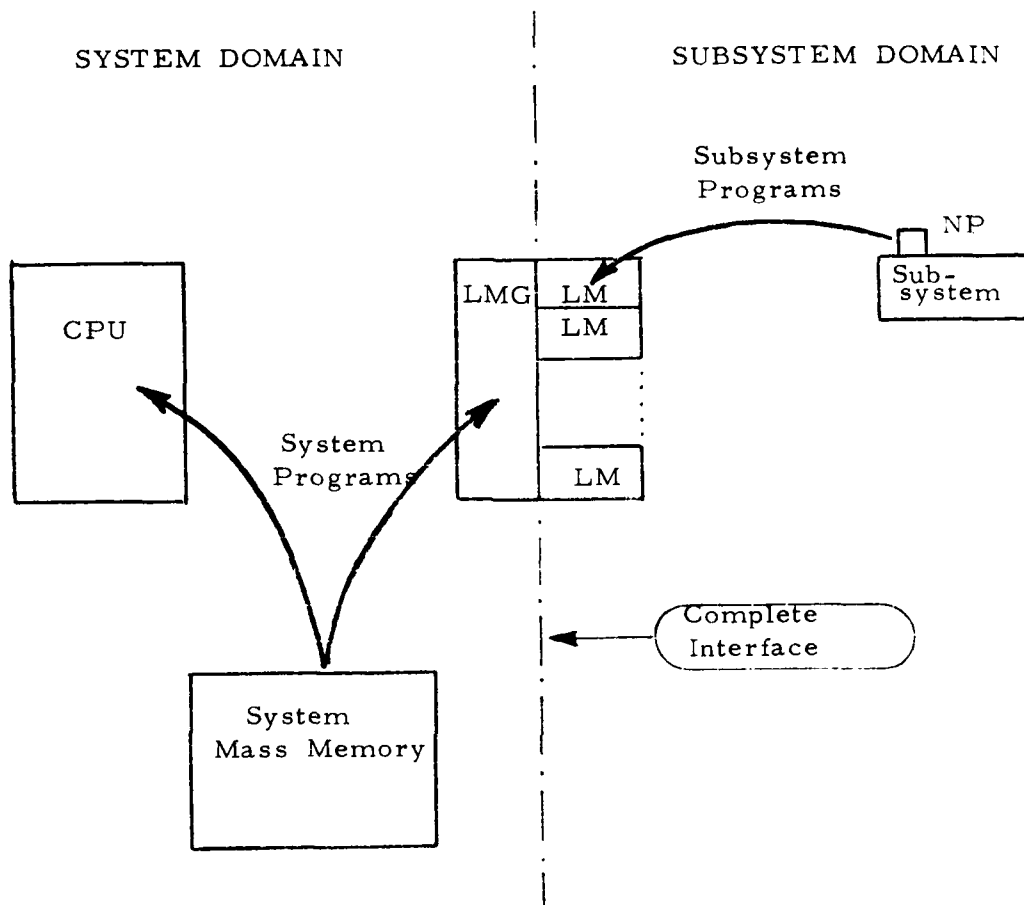


Figure 19 Storage and Loading of System and Subsystem Application Programs

for the CPU and the link manager of RLU's are downloaded from a system mass memory. These programs are aircraft-dependent and mission oriented. Subsystem programs complement the subsystem hardware in generating information that is free of device peculiarities. These programs are uploaded from a subsystem nameplate into the interfacing link module.

Table 5 identifies the maintenance and fault tolerance programs which are resident in the link manager or are downloaded from mass memory. Also in this table, software which is resident in a link module or which is uploaded from nameplates is identified. The interdependence between data conversion programs, maintenance programs and fault tolerance programs is illustrated in Figure 20. Data to and from a subsystem is processed by a data conversion program which contains monitoring routines for detecting subsystem failures. The detection of a fault by a monitoring program in the link module generates a flag to the fault management program in the link manager. The fault management program takes further action in establishing the nature of the fault and either promotes recovery or a shutdown of the subsystem. The fault management program may run a subsystem operational quality program (developed by the subsystem manufacturer) to determine a go/no-go status for the link module and its related subsystem. When the nature of the fault detected by the fault monitoring program clearly identifies a malfunction

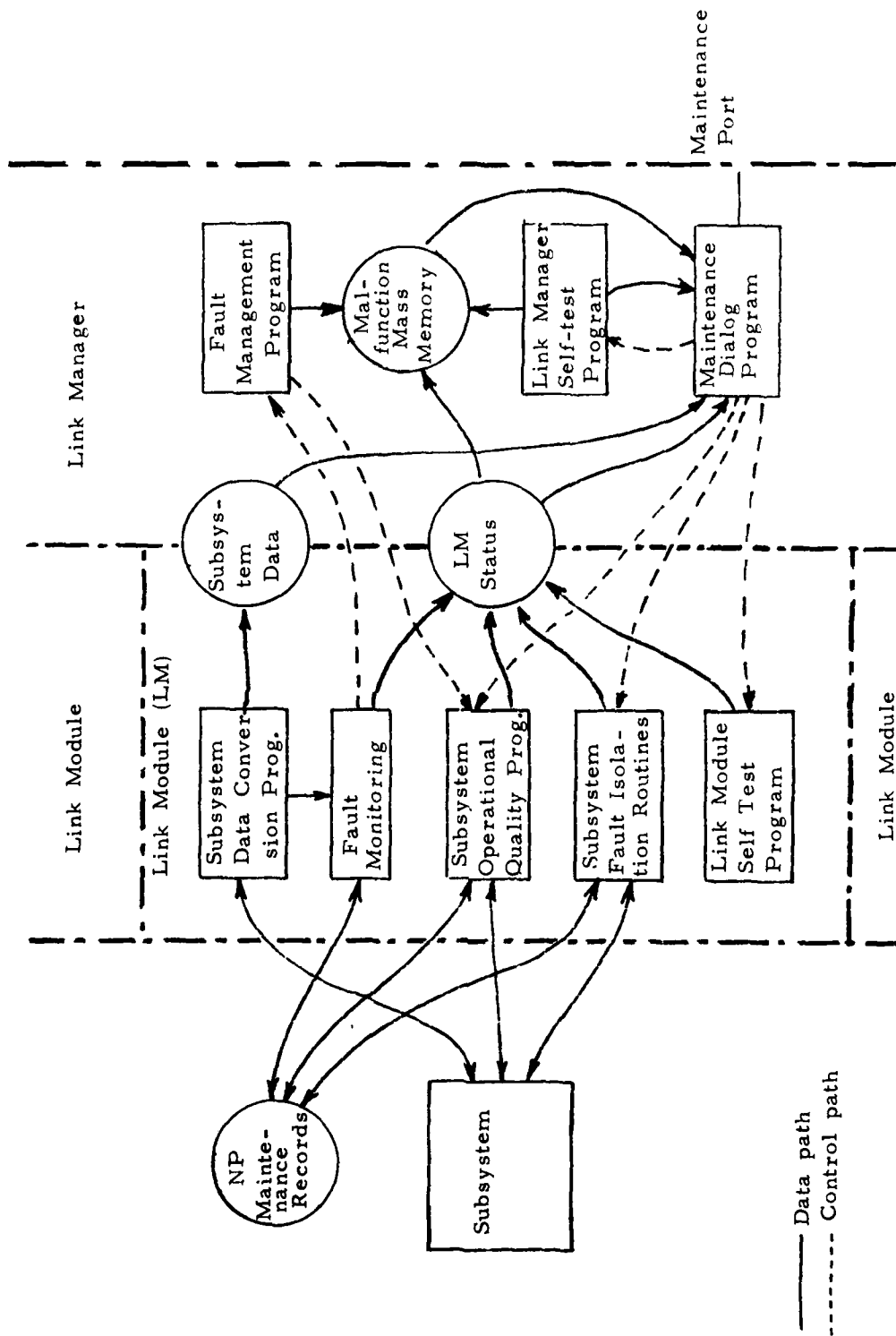


Figure 20 Flow of Control and Data Among Maintenance and Fault Tolerance Programs

TABLE 5

MAINTENANCE AND FAULT TOLERANCE SOFTWARE

Link Module

- a) Uploaded from subsystem nameplate
  - Fault Monitoring Program -- works as part of data conversion program to detect data errors or subsystem failures.
  - Fault Isolation Diagnostic Program -- stand alone program used to identify faulty LRU and SRU subsystem components.
  - Operational Quality Program -- stand alone program used to establish a go/no go status of the ensemble consisting of the link module and subsystem.
- b) Resident in LM firmware
  - Link module self-test -- stand alone program used to exercise all features of the link module and verify its operational integrity.
  - Fault recording routine -- service provided by the link module executive allowing failure information to be recorded on a subsystem nameplate.

Link Manager

- a) Downloaded from system mass memory
  - Fault Management Program -- stand alone program that is executed whenever a fault is detected in a link module or when the link manager becomes isolated from the system bus.
- b) Resident in LMG firmware
  - Link Manager Self-test -- stand alone program used to exercise all features of the link manager and verify the operational integrity.
  - Maintenance Dialogue -- This program interacts through the RLU's maintenance port with a portable CRT terminal. The terminal provides a maintenance technician with access to link manager and link module information and with control of diagnostic program execution.

in the subsystem, the link module will record the malfunction and its time of occurrence in the subsystem's nameplate. Each time the fault management program in the link manager is activated, an entry is made into the link manager's malfunction mass memory. This entry identifies the cause of activation and the operational environment at the time of its occurrence. The fault management program should be specified and designed under the supervision of the system designer so that the management of failure is consistent with mission objectives. The fault management program within the link manager should also schedule the execution of programs which support a stand-alone operation of the RLU as an independent cluster in the event that the RLU becomes isolated from the system.

Malfunctions within an RLU or any of the subsystems it interfaces with are recorded in the RLU's malfunction mass memory. The storage of information in this media is temporary for the duration of a mission and should be retrieved at its completion for evaluation of malfunctions and the scheduling of repair if needed. The retrieval of information from mass memory may be routed to the system CPU through the multiplex bus or to a maintenance console through the RLU's maintenance port. The latter option allows maintenance personnel to interact with the RLU and its interfaced subsystems. This is accomplished with the aid of

a maintenance dialog program which is resident in the link manager. The dialog program provides control of the execution of the link module self- tests and subsystem fault isolation routines. These diagnostic routines will indicate to the maintenance technician which system LRU needs to be replaced.

## 5.2 DATA STORAGE FOR MAINTENANCE

A structure for storage of malfunction data in an avionic system is illustrated in Figure 21. Two types of storage are indicated in the figure: malfunction mass memories and electronic nameplates. The nameplate is a device-related storage which contains information on maintenance and malfunctions which are specific to the device. The malfunction mass memory on the other hand contains a more varied type of information. It contains records of device malfunctions as well as of failures which are not directly traceable to a specific device, such as a report on inconsistent or unreliable data. The malfunction mass memory provides two functions: storing a list of all malfunctions within the RLU, and providing a redundant record of subsystem malfunctions in the event that subsystem nameplates have malfunctioned.

The malfunction mass memory stores information for individual missions and should be retrieved at the end of

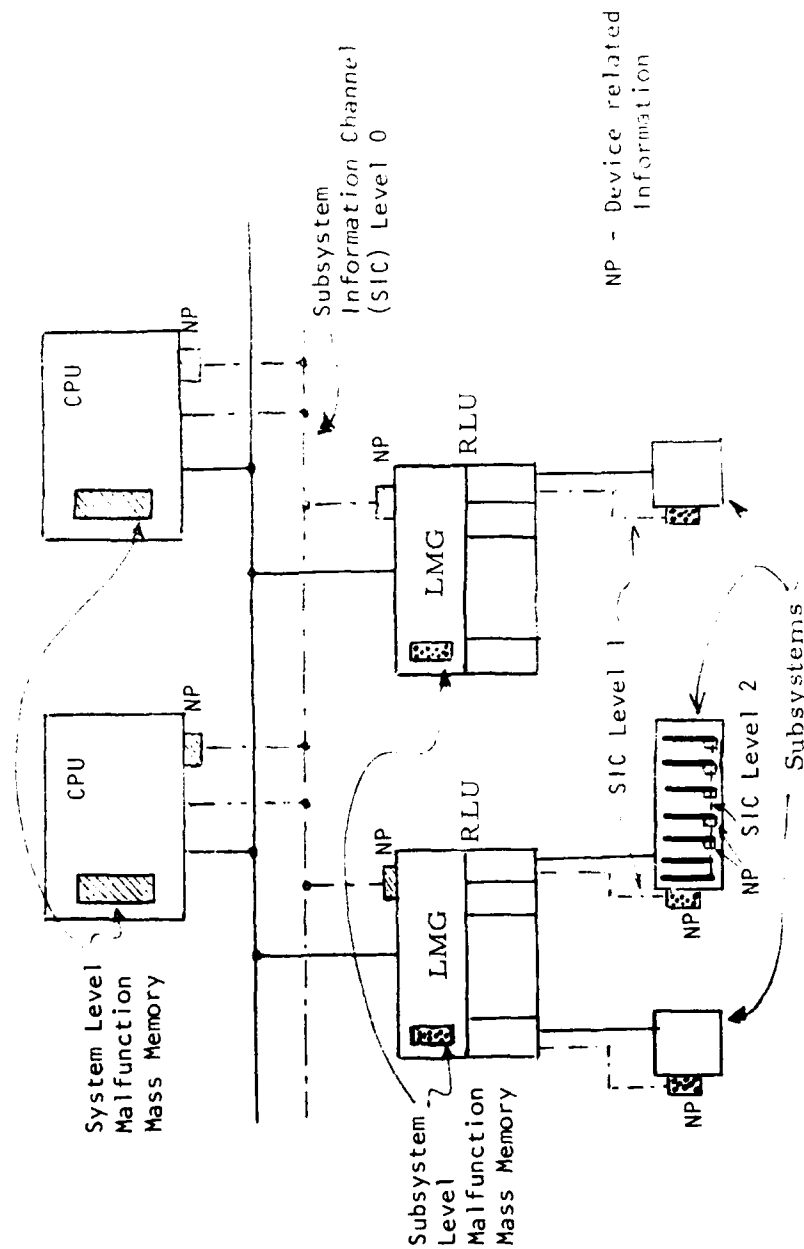


Figure 21 Hierarchical Structure for Storage of Malfunction Data

each mission and stored in a centralized maintenance facility data base for maintenance evaluation and collection of failure statistics.

### 5.3 PROGRAMMING LANGUAGES

The nature of the languages used in the implementation of programs for the link manager and the link modules may be quite distinct. The link manager object code should provide top performance execution and follow the standard used for the other system core processors. The link module, on the other hand, is designed to execute subsystem related programs and should therefore accept programs from a variety of sources. A universal object code that provides for execution of the link module programs by a machine-independent interpreter should be used. This interpretive object code should be compiled or translated from a high-order language which is easy to program and document. The selection of a universal interpretive language capable of supporting the processing requirements of link modules should be standardized. Among high-order languages, the following produce interpretive codes: BASIC, PASCAL, APL, and FORTH. All of these languages provide machine independent interpretive object codes which are efficiently executed and which may be transported among distinct processors equipped with the appropriate interpreter. Since ADA has been

selected by the Department of Defense as the common, standard high-order language, effort should be devoted to the development of an interpretive object code (similar to PASCAL's P-code) for storing subsystem programs in electronic nameplates.

## SECTION VI

### IMPLEMENTATION PLAN

This section describes a plan which provides for an orderly transition from current avionic system design to RLU-based system design. The plan calls for a gradual incorporation of RLU concepts and components such that maximum utility along with maximum flexibility is realized at each phase of the implementation. Thus, under this plan a delay in the development of one system component or concept will not jeopardize the implementation of avionic systems. The three phases of the plan envisioned by the authors are:

1. To incorporate nameplates and standardized signal interfaces into all new subsystem designs and develop Link Modules and Link Managers as stand-alone test and maintenance processors.
2. Replace remote terminal units with RLU's in those subsystems which have the required nameplates and interfaces. This phase would provide most of the RLU/Subsystem features presented in this report
3. Incorporate RLU concepts into an overall design-for-repair system architecture by adding RLU components (nameplates, maintenance mass memories, SIC's and SDC's) to the system CPU and RLU's. The

result will be an avionics system which can be tied into a centralized maintenance computer facility for automated system checkout.

Each of the above three phases will consist of two basic cycles:

1. Definition of standards
2. Procurement/Implementation

#### 6.1 PHASE I - SUBSYSTEM INTERFACE STANDARDIZATION

Phase I includes the development and partial utilization of LRU nameplates and the standardization of signal interfaces throughout the subsystem.

Placing nameplates on the LRU's allows a Link Module to be used during the production troubleshooting and checkout of the LRU. This requires a parallel effort to develop a first generation RLU to be used in this fashion, an effort which will prove very useful during Phase II. Note that the ability to produce avionics systems with this approach is not jeopardized as the LRU will be usable in a conventional system using RTU's even if the development of the supporting RLU components is delayed. In fact the LRU will continue to be interfaced through RTU's in the present fashion except that the signal interfaces will be designed to be compatible with an ICA/LM.

As the ICA/LM is developed it may be used during production checkout of the LRU. This testing will make use of the nameplate to store diagnostics and test results. The LM/RLU can be used in a portable configuration as a maintenance aid. This would be similar to the portable use of an RLU as described in the Appendix B ("Retrofitting"). This application will bring the RLU into the maintenance area as the RLU is developed and tested while allowing development of LRU's to proceed independently.

The use of nameplates and standardized interfaces will have minimal effect on the mechanical properties of subsystem LRU's. We anticipate that the standardized interface signal conventions will allow the use of a standard connector on all LRU's. The standard connector should provide the capability of controlling and monitoring the subsystem to the SRU level. The nameplate is envisioned as a small unit applied to the outside surface of an LRU having its own unique connector for use on the SIC of an LM. The use of standardized connectors for LRU signal interfaces will aid maintenance and repair crews even in the absence of on-board RLU's. Crews may also use portable RLU's as checkout and maintenance processors as the on-board RLU's are being developed and tested.

LRU's designed under the guidelines of Phase I will continue to be interfaced to the on-board system through RTU's. This requires that the computational capabilities

which the LM will ultimately provide must be incorporated into the subsystem's nameplate. It may be desirable to design an LRU so that it may be replaced at a later time when the capabilities of the LM are available on-board. Notice that some of the advanced self-testing features of an LRU/LM combination will not be available under the Phase I guidelines.

## 6.2 PHASE II - RLU INTEGRATION

Phase II allows for the incorporation of RLU's into those subsystems whose LRU's had been designed with electronic nameplates and standard interfaces. Note that RT's and RLU's could coexist in a system during this phase.

For those subsystems utilizing RLU's, many of the features presented in this report may be realized. The RLU, working with nameplate software, will provide continuous monitoring capability, assist in the repair and maintenance operations, interact extensively with interfaced LRU's, etc.

Phase II also provides an opportunity to develop the techniques of functional clustering and the other aspects of fault tolerant design described in Section 6.

As Phase II is implemented, the LRU design process must evolve to take advantage of LM capabilities. LRU simplification, resulting from a utilization of LM signal scaling and computation, should evolve during this phase. The

development of techniques for on-line monitoring and fault recording should also occur during Phase II.

We anticipate that the use of the RLU maintenance port for repair operations will be perfected during Phase II. This would be a logical extension of using portable RLU's as test equipment (see subsection 6.1).

### 6.3 PHASE III - AUTOMATED SYSTEM MAINTENANCE

This final phase enables the extension of RLU concepts to all levels of an avionic system.

The proposed relationship between any two levels in a system is indicated in Figure 22. This represents a generalization of the relationship which exists at the RLU/Subsystem interface. Each level in the system would communicate with lower levels over two distinct data paths:

1. The conventional information interface (SDC)

and

2. A maintenance oriented interface which allows nameplate information to flow between levels (SIC).

Each level would have the functional equivalent of a Link Module, a Maintenance Port as described for the RLU, and a maintenance oriented mass memory.

By incorporating the above capabilities at all levels in a system, the features supported at the RLU/Subsystem interface would be available throughout the system.

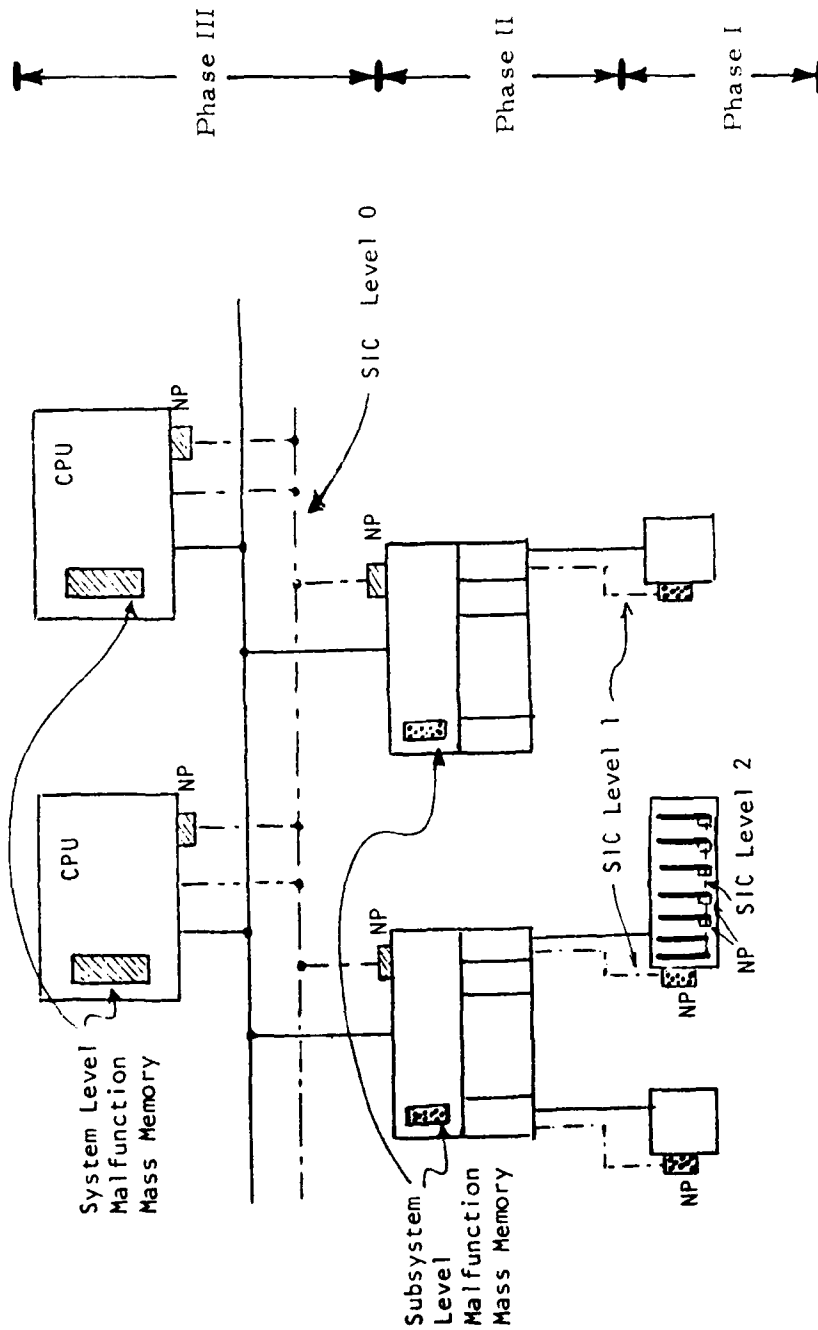


Figure 22 Design-for-repair System Architecture

The inclusion of the above components will facilitate the use of a centralized maintenance computer facility. The centralized facility may interface to the system at a higher level and have access to the nameplates of all system components including the system CPU's. This will allow automatic testing of the complete architecture shown in Figure 22 from the centralized facility.

#### 6.4 IMPLEMENTATION PLAN SUMMARY

This section has described a plan which gradually incorporates RLU concepts into avionic systems. The three phases of this plan are designed to be largely self-sufficient. If Phase I is carried out and a decision is made to delay the implementation of Phase II, the benefits from Phase I would still be substantial. The same is true of Phase II relative to Phase III. Figure 23 outlines the activities of the proposed implementation plan presented in this section. We feel that this implementation plan will facilitate the maximum utilization of the potential of the RLU if it is carefully and thoughtfully executed.

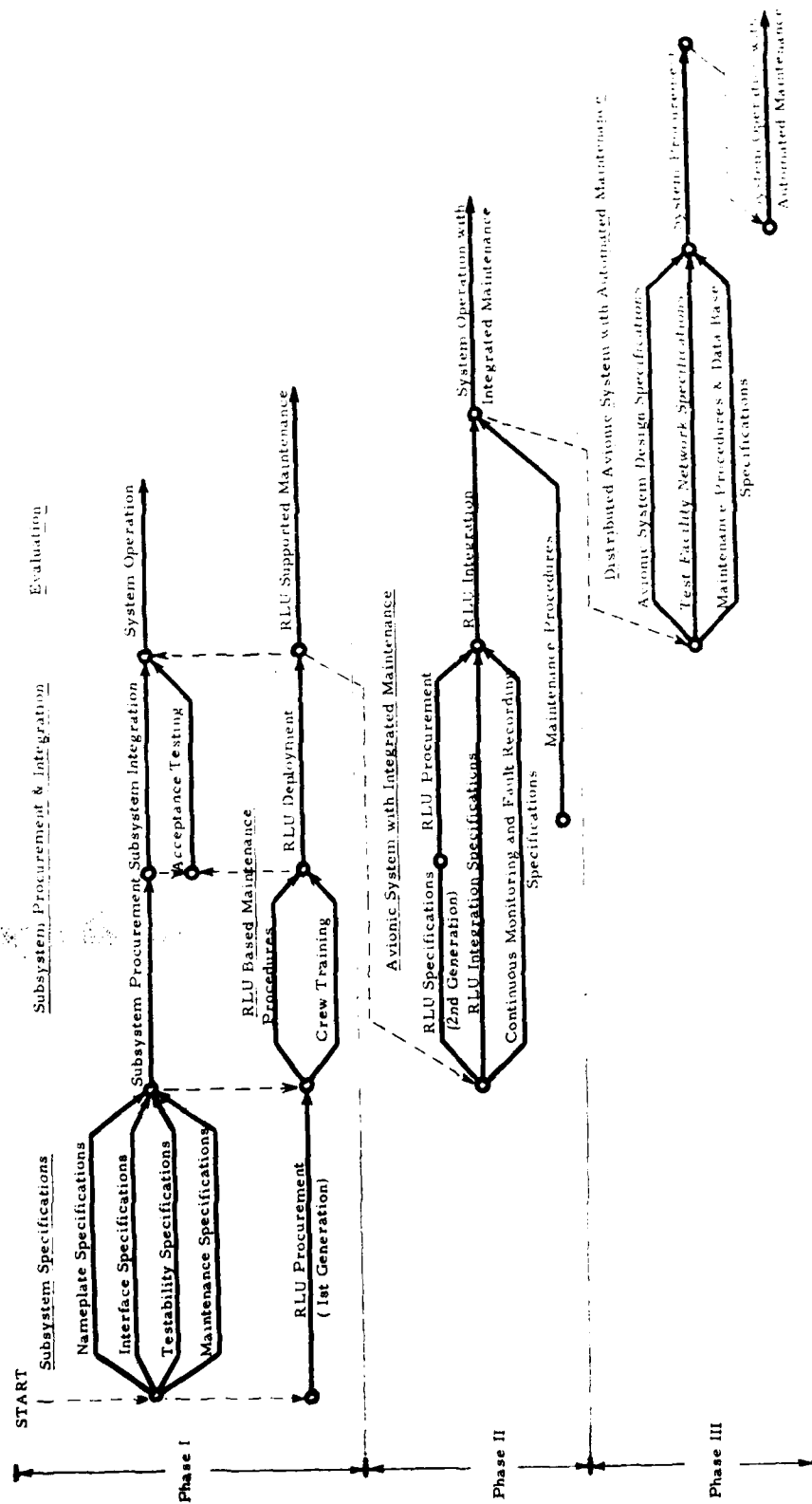


Figure 23 Time Phased Implementation of RLU Based Avionic Systems

## SECTION VII

### SUBSYSTEM DESIGN SPECIFICATIONS

This section provides a general description of the necessary specifications required for subsystems of RLU based avionic systems. It will thus provide a description of the subsystem manufacturer's responsibilities in the development of hardware and software for the support of automated maintenance.

As described in Section 6, the incorporation of RLU's into avionic systems will be gradual; therefore subsystems should be specified to operate with both RT's and RLU's. As usual, each subsystem will be characterized by both physical and performance specifications. The physical specification of each subsystem will require the addition of an electronic nameplate. Performance specifications of each subsystem, however, will be significantly affected by the use of RLU's in avionic systems. Five specifications must be provided for RLU compatible subsystems. These are respectively:

1. interface,
2. testing,
3. operational,
4. program, and
5. nameplate.

## 7.1 INTERFACE SPECIFICATIONS

One interface specification will require the usage of the standard link module interface connector for all newly incorporated subsystems. Other interface specifications will describe how the subsystem is connected to RT's or to RLU's. If the subsystem circuit used in the interface with an RT is distinct from that used in the interface with an RLU, the control logic for selection of the appropriate circuit shall be provided in the interface specifications.

The interface specifications will also include a clear description of information to be supplied to the subsystem (control data), or to be acquired through the subsystem (monitoring data). The information specification should identify the variables that are measured or controlled by the subsystem, as well as their engineering units and data formats. In addition, the accuracy and the reliability of the measured data should be part of the information provided by a subsystem to the CPU. The information (along with its value, accuracy, and reliability) should arrive at the CPU via the link module's shared memory.

The interface specifications must also include a description of the controls for the selection of modes of operation, and status for monitoring the operation of hardware, software, and data transfer.

## 7.2 OPERATIONAL PERFORMANCE

A value for the mean time between failure (MTBF) will be provided for each subsystem for various environmental conditions (acceleration, temperature, voltages, noise, etc.) as specified by the appropriate MIL-STD. The operational performance specification shall describe the data processing necessary to convert subsystem dependent data into hardware independent information. The subsystem and the link module shall support the collection of statistical data such as the time of operation required to establish a correct measured figure of an MTBF under diverse environmental conditions. The statistical information shall be stored in the subsystem's nameplate to provide a check on warranted components. This feature could be extremely important in the evaluation of modules for the military computer family (MCF)[8]. Such modules are designed to be interchangeable among machines and have strict warranted MTBF's which are monitored for warrantee compliance. The collected statistical data will greatly assist subsystem manufacturers in the correction of operational problems.

## 7.3 TESTING SPECIFICATIONS

Each shop replaceable unit (SRU) within a subsystem shall be controllable and observable through the link module. The ensemble consisting of a link module, one or more

subsystems, and the associated connectors shall allow the detection and identification of failures at the SRU level. The design of hardware and software for the detection and isolation of faults is the responsibility of the subsystem manufacturer. This will not constitute a new burden for the manufacturer since he normally must design programs along with a test set-up for use at the factory level. Such set-ups are used by production personnel to repair units which do not meet quality assurance (Q.A.) specifications.

The subsystem designer shall develop an online monitoring program to be processed by the link module for the detection of malfunctions in any of the elements of the ensemble. Detection of a failure by the fault monitoring program will result in a flag being set and will cause the link manager to schedule execution of the fault management program.

#### 7.4 PROGRAM SPECIFICATIONS

Three programs are required for each subsystem and should be stored in the subsystem's nameplate. These programs (described in Section 5) are:

1. A data conversion program.
2. A subsystem operation quality program.
3. A subsystem diagnostic program.

AD-A086 126

HOUSTON UNIV TX DEPT OF ELECTRICAL ENGINEERING

F/S 1/3

THE REMOTE LINK UNIT: APPLICATIONS TO THE DESIGN-FOR-REPAIR NET--ETC (11)

APR 80 C J TAVORA, M A SMITHER

F33615-78-C-1634

UNCLASSIFIED

AFMIL-TR-80-1033

NL

2-1-2

20  
200000

END  
DATE  
FILMED  
8-80  
DTIC

The data conversion program allows for the conversion of raw data from a sensor into information consisting of a value, its accuracy, and reliability. Likewise, data from the CPU is converted into electrical signals compatible with the range and accuracy of the actuator being driven. Embedded in the data conversion program is a fault monitoring program as described in subsection 7.3. The subsystem operational quality program, which is designed for use at the factory by Q.A. personnel, can also be used during system operation to validate the operational status of the ensemble consisting of the link module, cables/connectors, and the subsystem. This program shall be used to establish a go/no-go status for the subsystem. The subsystem diagnostic program supports the detection, isolation, and identification of any malfunctioning SRU within the subsystem.

#### 7.5 NAMEPLATE SPECIFICATIONS

Each subsystem shall be delivered with an electronic nameplate containing the following subsystem related data: identification, interface configuration, programs, and operational records.

The identification section of the nameplate contains information such as: device type (name and military classification), model, and serial number. The interface configuration specifies the interface connections and signals

required for subsystem operation with an RT or an RLU. The connector pin assignment including each signal name, signal type, and timing and handshake information shall be provided. The programs required for data conversion, online monitoring, fault detection, and operational quality evaluation shall be listed in a directory.

A section of nonvolatile read/write memory shall be provided in the nameplate. This shall be used for storage of data records of subsystem failures, the origin of the recording, and the subsystem's operational environment.

The nameplate shall also store records describing repair or calibration performed on the subsystem and any modifications resulting from a design update of the subsystem.

APPENDIX A

THE REMOTE LINK UNIT - AN ADVANCED  
REMOTE TERMINAL CONCEPT

IEEE Transaction on Industrial Electronics  
and Control Instrumentation

August 1979

# The Remote Link Unit—An Advanced Remote Terminal Concept

CARLOS J. TAVORA, SENIOR MEMBER, IEEE

**Abstract**—Remote terminal units are extensively used in digital automation systems to interface a central processing unit (CPU) to remotely located sensors and actuators. This paper presents a conceptual description of the remote link unit (RLU) which overcomes inherent limitations of remote terminal units. The RLU utilizes a single type of interface card to support most process I/O interfaces. The RLU identifies sensors, actuators, and subsystems through the use of electronic nameplates. Each nameplate provides information regarding device function, location, interface parameters, and calibration status. In addition it stores programs for device handling, engineering unit conversion, and device diagnosis. The RLU allows sensors and actuators to be relocated or substituted without requiring changes in the CPU software to correct device addresses or conversion constants. The electronic nameplate may also be used for inventory control, automatic calibration, and maintenance of devices.

## 1. INTRODUCTION

REMOTE terminal units (RTU) are used extensively by designers of digital data acquisition and control systems to interface the central processing unit (CPU) with devices which are not close enough for connection to the I/O bus [1]–[7]. The remote terminal, as the name implies, provides termination for computer signals at a remote location through specialized termination cards (interface modules). Communication between the CPU and a remote terminal is implemented through serial digital transmission [8]–[10]. A controller card at the remote terminal carries out all functions required to support the operation of interface modules. These functions include message reception and transmission, serial to parallel format conversion, communication error detection, message interpretation, and timing and control of data transfer to and from interface modules. When interfaced through a remote terminal, a device is identified by the subaddress location of the interface module and the channel through which it is connected. This arrangement requires the use of address tables associating subsystems with their corresponding interface modules and channels. A meticulous and precise address table modification procedure must therefore be carried out whenever a change in system configuration takes place.

The remote terminal illustrated in Fig. 1, supports CPU interfaces to a thermocouple, a heater control, and a display

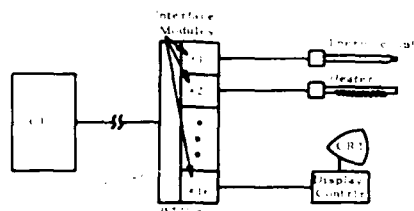


Fig. 1. Remote terminal interfaces to a thermocouple, a heater control, and a display subsystem.

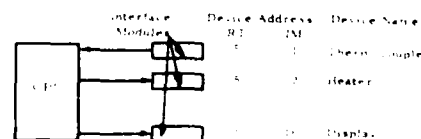


Fig. 2. The thermocouple, heater control, and display subsystem are seen by the CPU as interface modules of a remote terminal.

generator. Typically, one interface module is required for each interfaced device. The address configuration of these subsystems as seen by the CPU is depicted in Fig. 2. Data transfers to or from each device are accomplished with commands which have distinct formats and which reference different addresses. If, for example, the interface modules used with the thermocouple (A/D card) and the heater control (D/A card) are interchanged or relocated to another remote terminal, the CPU will not be able to communicate with either device. One may conclude that remote terminal interfaces require perfect coordination between the assignment of devices to interface modules and the corresponding entries in the device address table. This requirement is a nuisance to the technical personnel who must carry out tests utilizing digital data acquisition and control systems to perform automated test on test articles requiring a large number of sensors and actuators. Each time reconfiguration must take place either as a result of a test modification, or to correct the malfunctioning of a sensor or interface device, modifications must be concurrently made on the software device address tables to reflect the changes in system configuration. This not only requires careful planning and data base manipulation, but involves a considerable amount of documentation which is time consuming and is susceptible to human error.

An additional limitation of a typical remote terminal is the requirement that it must support several different types of interface modules [3]–[6]. Analog input and output, serial or

Manuscript received December 18, 1978; revised April 26, 1979. This work was supported initially by the University of Houston under Grant NROF-EE-K-91. A study of the feasibility of implementation in avionics systems is being supported by the U.S. Air Force Systems Command, Aeronautical Systems Division under Contract F33615-78-C-1634.

The author is with the Electrical Engineering Department, University of Houston, Houston, TX 77004.

TABLE I  
FEATURES OF THE REMOTE LINK UNIT (RLU)

1. AUTOMATIC IDENTIFICATION OF INTERFACED INSTRUMENTATION	
•	RETRIEVES INFORMATION FROM THE ELECTRONIC NAME-PLATE OF AN INTERFACED DEVICE
•	MAINTAINS A DIRECTORY OF INTERFACED DEVICES WITH NAME DESCRIPTOR AND LOGICAL UNIT NUMBER
•	IDENTIFIES LOCATION AND FUNCTION OF EACH DEVICE WITHIN A SYSTEM
2. DEVICE INDEPENDENT I/O INTERFACES	
•	CONVERTS THE DATA FROM EACH DEVICE INTO A CONVENIENT DEVICE INDEPENDENT FORMAT
•	ALLOWS THE CPU TO ADDRESS DEVICES BY EITHER DEVICE IDENTIFIER OR LOGICAL UNIT NUMBER
3. UNIVERSAL INTERFACE MODULES	
•	ADAPTS THE INTERFACE SIGNALS TO SATISFY THE DEVICE SIGNAL REQUIREMENTS (LEVELS & TIMING)
•	USES A STANDARD CONNECTOR FOR ALL DEVICE INTERFACES
•	UPLOADS SPECIAL PROGRAMS (DIAGNOSTICS, DATA CONVERSION, AND DATA VALIDITY CHECK)
4. PROCESSING AT EACH INTERFACE MODULE	
•	PERFORMS SYSTEM LEVEL CRITICAL CONTROL FUNCTIONS IN CASE OF SYSTEM FRAGMENTATION
•	CONVERSION OF DATA FORMAT AND DATA VALIDITY CHECK
•	PERIODIC TESTING AND CALIBRATION OF INTERFACED DEVICES

parallel digital input and output, and many other interface signal formats are required to handle devices which might be connected through a remote terminal. Indeed, even within a single category such as serial digital input there can be many subcategories necessitating still more distinct interface types. The inventory and maintenance expertise necessary to support the large variety of interface modules required by distinct external devices poses a major logistics problem.

Still another limitation of the remote terminal is its lack of ability to operate stand-alone supporting control and monitoring function in a degraded mode without CPU assistance. Such capability is essential for operation critical processes which require well-defined shutdown sequences in order to avoid catastrophic losses in production or equipment as a result of system control fragmentation.

The problems associated with remote terminals can be traced to a basic operational concept which treats them as peripheral devices rather than as a transparent means for communication between CPU and devices. An objective review of the desirable features which should be provided by a general purpose remote interface between CPU and external devices, has led to the concept of a remote link unit (RLU) which is proposed in this paper. A summary of the significant features of the RLU is outlined in Table I.

The RLU provides a complete interface since, in addition to implementing data transfers, timing and control signals, it will also provide a channel through which the CPU can interrogate each external device for that device's identity. The RLU interface modules are transparent in the sense that the CPU addresses devices with which it desires to communicate using logical device numbers assigned during run time. This function may be accomplished by maintaining a dynamically

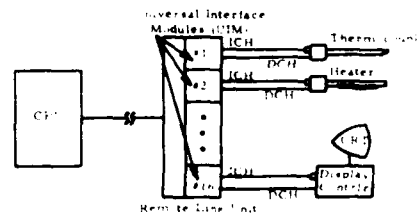


Fig. 3. Remote link interfaces to a thermocouple, a heater control, and a display subsystem.

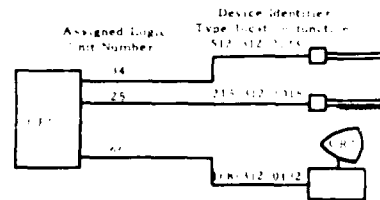


Fig. 4. The thermocouple, heater control, and display subsystem are seen by the CPU as independent peripheral devices. Each device has a unique identifier and a CPU assigned logical unit number through which it is addressed.

updated subsystem directory. When a command is received, the RLU checks its directory and proceeds to transfer the information to the corresponding subsystem. Since this directory is dynamically updated by the RLU to reflect the current external device configuration, it is possible to modify that configuration, e.g., move a device connection from one interface module to another, without modifying the CPU device tables. An illustration of the RLU interface to a thermocouple, a heater control, and a display generator is presented in Fig. 3. In this case these devices are seen by the CPU as individual and peripheral, each addressable as a logical device as shown in Fig. 4.

The RLU will support universal interface modules (UIM) for connection to external devices. Such interface modules will provide all types of signal interfaces (ac or dc analog, serial or parallel digital) on a single card which can be configured on demand to accommodate any device interface signal requirements. To accomplish this each UIM should have, in addition to the conventional data channel (DCH), a separate identification channel (ICH) for subsystem identification and interface requirements specifications as shown in Fig. 3. This channel should have an identical format for all devices. A communication protocol for the identification channel should be used to control the transfer of information between UIM and device.

The RLU should provide processing at the UIM thus allowing several functions to be performed at the remote location independent of CPU operation. This feature allows validity checks to be carried out on data originating at external devices prior to its transfer to the CPU. In addition, the proper operation of a subsystem may be monitored through its status and data. Format conversion and data limit checking may also be performed at the interface and thus reduce the CPU processing load. By allowing interface modules to have stand-alone processing capability, control and display functions may be executed at the RLU in case of system fragmentation. The re-

TABLE II  
CONTENTS OF THE ELECTRONIC NAMEPLATE

1. DEVICE IDENTIFICATION
• TYPE, MODEL AND SERIAL NUMBER
• SYSTEM FUNCTION
• SYSTEM COORDINATES (LOCATION)
2. SPECIFICATION OF INTERFACE SIGNALS
• SIGNAL LEVELS
• HANDSHAKE AND TIMING
• TRANSMISSION RATE
3. DEVICE OPERATIONAL RECORD
• CALIBRATION RECORD
• OPERATIONAL PERFORMANCE RECORD
4. DEVICE SUPPORT FIRMWARE
• DIAGNOSTIC PROGRAM FOR DEVICE TESTING
• INSTRUMENT CALIBRATION PROCEDURE
• DEVICE HANDLER
• DATA CONVERSION ROUTINES

remote link unit described in this paper is based on three major concepts which will be described next: the electronic nameplate, the universal interface module, and the link manager.

## II. THE ELECTRONIC NAMEPLATE

The electronic nameplate is a key concept which not only leads to the implementation of the remote link unit but also supports the automatic calibration, testing and inventorying of devices. As the name indicates, the electronic nameplate provides device identification by supplying type, model, and serial number. In this sense, the electronic nameplate supplies information similar to that provided by the universal product code (UPC) in packaging. However, the electronic nameplate will provide additional information which far exceeds the simple identification of a device. This information will include the following: device function and location as a part of a system, parameters specifying the type of interface signals required, records of the operational performance and calibration of the device, and, finally, programs (subroutines, routines, and tasks) which may be used to perform device diagnostic testing, device handling and data conversion.

By storing in each instrument the software required to convert its data into a device independent format, it is possible to design system programs that are device independent. Henceforth, similar instruments having distinct interface characteristics may be interchanged without effecting the CPU software. A summary of the possible contents of the electronic nameplate is presented in Table II.

A possible architecture for the electronic nameplate is illustrated in Fig. 5. An extension port on the device nameplate is used to interface a secondary nameplate. This port is used to retrieve information related to the device's function and location as part of a system from a nameplate installed in the system's frame where the device is mounted. In addition to the interface port, the electronic nameplate should include a read-only memory containing the device identification and interface parameters, as well as device handlers and other programs which may be retrieved by the RLU. An electrically

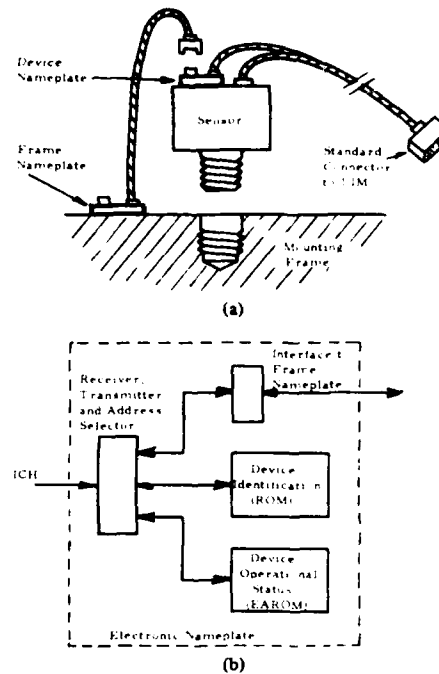


Fig. 5. The electronic nameplate is distributed among the device and the mounting frame (a). The internal architecture of the nameplate includes a controller, two interfaces, and memory (b).

alterable memory should be provided for storing the device calibration and operation records. Transfer of information from the electronic nameplate to the universal interface module may be implemented with a serial data communication controller powered by the interface module.

The electronic nameplate may find extensive use in the automatic calibration, maintenance and inventory control of external system devices.

## III. THE UNIVERSAL INTERFACE MODULE

The universal interface module (UIM) will support an interface for any device which can be configured with the assistance of an electronic nameplate. The interface module connector will have a preestablished pin assignment which may support several distinct functions through the same set of pins. The exact configuration of the pins in terms of data signals and control signals is specified from the configuration parameters obtained from the electronic nameplate. Initially, all lines of the universal interface connector are maintained at a high-Z state. The universal interface module, upon detecting the connection to an external device will proceed to interrogate the electronic nameplate and retrieve its data. The device identification parameters will be the first retrieved. These will be followed by the interface requirement parameters which are used to select the appropriate data and control signals to the interface connector. The electronic nameplate will be interrogated for the existence of a diagnostic program. This program should be uploaded to the interface module for execution to establish the operational status of the interface module and

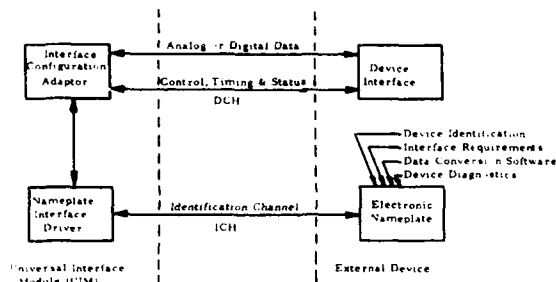


Fig. 6. Block diagram of the interface between a universal interface module and an external device.

TABLE III  
CHARACTERISTICS OF THE UNIVERSAL INTERFACE MODULE (UIM)

1. STANDARD SENSOR/ACTUATOR INTERFACE
• UTILIZES A STANDARD CONNECTOR
• SUPPORTS INTERFACE TO THE ELECTRONIC NAMEPLATE (IDENTIFICATION CHANNEL)
• ADAPTS THE INTERFACE TO SATISFY SENSOR/ACTUATOR SIGNALS (DATA CHANNEL)
2. IDENTIFICATION CHANNEL
• DETECTS DEVICE PRESENCE
• IDENTIFIES INTERFACED DEVICE
• MAINTAINS DEVICE OPERATIONAL RECORD
• RETRIEVES PROGRAMS FROM THE NAMEPLATE
3. MODULE PROCESSING
• DEVICE INDEPENDENT PROCESSING
INITIALIZATION
PERIODIC SCAN
MODULE DIAGNOSTIC
• DEVICE DEPENDENT PROCESSING
HIGH LEVEL INTERPRETIVE LANGUAGE
DATA CONVERSION
DEVICE DIAGNOSTIC
4. INTERFACE TO LINK MANAGER
• DEVICE INDEPENDENT FORMAT
• DEVICE STATUS INFORMATION (DIRECTORY)
• MODULE HARDWARE CONTROL AND STATUS
• MODULE TASK CONTROL AND STATUS
• INTERMODULE AND CPU COMMUNICATION

the device. The retrieval of data conversion programs from the electronic nameplate into the interface module will allow data to be maintained in a device independent format. The interface elements of an UIM and a device are shown in Fig. 6.

In order to maximize the flexibility of the universal interface module, a general purpose interpretive language should be used at the interface module for processing data and controlling the device. The use of an interpretive language will make the device programs processor independent. An interpretive language such as APL or Basic may be used as such a machine independent language. The device data processed by the interface module is transferred to the CPU through the RLU's link manager which communicates with interface modules through shared memory. This shared memory may also be used to transfer processing commands to the interface modules, store the operational status of the device, and maintain all information pertinent to the RLU device directory such as device type, function, location, and logical unit number. A summary of the features provided by the universal interface module is presented in Table III.

#### IV. LINK MANAGER

The link manager provides the interface between the CPU and universal interface modules. The functions performed by the link manager include the upkeep of the directory of interfaced devices, the control of data flow between the CPU and universal interface modules, the control of the RLU hardware resources, and the coordination of local processing among the universal interface modules to support stand-alone RLU operation.

The directory of interfaced devices contains the names of all devices which are interfaced through the RLU. The directory contains the location of the universal interface module through which the device is interfaced and possesses a logical unit number assigned to the device by the CPU. In this manner, a cross reference between physical location of an interface module, the name of the device it interfaces, and the corresponding logical unit number is established. In addition to the identification, the directory provides information relevant to the operational status of each device and its corresponding universal interface module. This information should be generated through the execution of diagnostic programs which exercise the interface module and the external device. The RLU directory should also provide identification and execution status information on all external programs loaded into an interface module. These programs may have been uploaded from a device to the universal interface module or downloaded from the CPU. Whenever the system configuration is modified either to correct a device failure or to alter the scope of the system functions, an update of the system directories takes place. When a change of configuration occurs, all affected remote link units update their directories and the CPU is informed that it should update its copy of the directory tables so as to reflect the correct system configuration. In normal operation, the link manager updates the directory of interfaced devices based on the data provided by the universal interface modules. When the system is initialized, the CPU should interrogate each link manager in order to obtain the name descriptor of all interfaced devices, following which the CPU issues a logical device number to each device for addressing during normal system operation. The link manager identifies the address tag (logical device number) of data from the CPU and routes it to the corresponding universal interface module. In this sense the link manager performs the same functions as a front-end processor used as a data concentrator for a multiterminal environment.

The link manager controls the hardware resources of the RLU such as redundant power supplies, alternate communication channels, and available memory in order to maintain its operation in the eventuality of failure of any one of these components. As such, it can control these resources based on its internal status or as a result of commands received from the CPU. The link manager will monitor the system communication to decide when it is isolated from the CPU, in which case it will coordinate intermodule processing for stand-alone operation. This will allow critical control functions to be processed locally such as an orderly shutdown of devices and functions which cannot operate when the remaining system is isolated. A summary of the functions provided by the link manager is

TABLE IV  
FUNCTIONS OF THE LINK MANAGER

1. MAINTENANCE OF RLU DIRECTORY
• DEVICES (NAME, FUNCTION, LOCATION, STATUS)
• MODULES (HARDWARE STATUS, SOFTWARE STATUS)
• RLU (COMMUNICATION STATUS, PROCESSING FUNCTIONS, HARDWARE STATUS)
2. COMMUNICATION TRAFFIC CONTROL
• TRANSFER OF DATA AND COMMANDS BETWEEN CPU AND UIM'S
• COMMUNICATION ERROR RECOVERY
• DETECTION OF RLU ISOLATION FROM SYSTEM
3. COORDINATION OF RLU PROCESSING
• CONTROL OF SYSTEM LEVEL TASK PROCESSING AMONG UIMs (RLU STAND-ALONE OPERATION)
• MANAGEMENT OF MULTIPLE REDUNDANT SENSORS
• SYSTEM POWER-UP AND SHUT-DOWN SEQUENCES
4. CONTROL OF RLU HARDWARE RESOURCES
• SELECTION OF RLU POWER SOURCE
• SELECTION OF COMMUNICATION BUS
• ISOLATION OF DEFECTIVE RLU MODULES

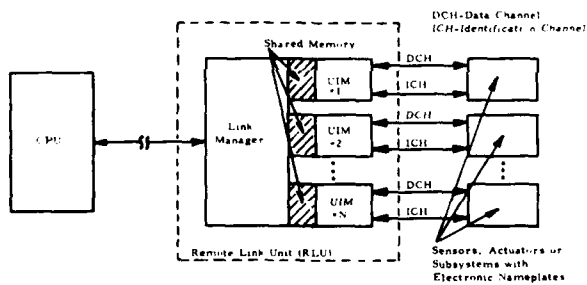


Fig. 7. Components and interfaces of the remote link unit.

outlined in Table IV. The architecture of the remote link unit in terms of the link manager and the universal interface modules is presented in Fig. 7.

## V. CONCLUSIONS

This paper has presented the concept of a remote link unit which incorporates several enhancements not available in existing remote terminal units. The most significant concepts introduced are the electronic nameplate and the universal interface module. The electronic nameplate will significantly simplify the design of software for systems which must be re-configured dynamically, and will facilitate the checkout and

calibration of devices. By providing devices with a complete name descriptor which includes type, function, and location, it is possible to allow the central processor to establish whether or not a required device is interfaced and to address it through a logical unit number. Also, by allowing the electronic nameplate to contain data conversion programs, it is possible to perform data conversion at the interface modules so that the device data is available to the CPU in a device independent format (such as floating point engineering units). This technique will greatly simplify the replacement of sensors and actuators in most test facilities and will make the device transparent to the programmer.

The concept of a universal interface module capable of supporting all external device interfaces and utilizing a standard connector will uncomplicate the interfacing procedures and significantly decrease the inventory of cards required to support process control interfaces. The capability of local processing at the interface module with programs which are either uploaded from a device or downloaded from the CPU, provides both flexibility and fallback capability.

The concepts presented in this paper may be integrated to implement the remote link unit or may be individually added to existing remote terminal units in order to enhance their operation as system components.

## ACKNOWLEDGMENT

The author wishes to express his gratitude to T. A. Jagen, Jr., for the helpful discussions and suggestions.

## REFERENCES

- [1] D. Horelick and R. S. Larsen, "CAMAC: A modular standard," *IEEE Spectrum*, vol. 13, pp. 50-55, Apr., 1976.
- [2] *MODAC, Reference Manual*, Modular Computer Systems, Inc., 1975.
- [3] *RTP-Process Interface Subsystems, User Manual*, Computer Products, Inc.
- [4] *TRW 2000 Remote Terminal Unit, Reference Manual*, TRW Controls.
- [5] *Series 5000 Remote, Instruction Manual*, Harris Controls, 1978.
- [6] *Input/Output Interface Subsystems Model 11XX, Tech. Manual*, Modular Computer Systems, Inc., 1975.
- [7] C. W. Rose, E. Linn, and J. D. Schoeffler, "Distributed microcomputer data acquisition," *Instrumentation Technology*, vol. 20, pp. 55-61, Dec. 1975.
- [8] J. Washburn, "Communications interface primer-Part I," *Instruments and Control Systems*, pp. 43-48, Mar., 1978.
- [9] —, "Communications interface primer-Part II," *Instruments and Control Systems*, pp. 59-64, Apr., 1978.
- [10] J. D. Schoeffler, "Data highway structures and their effects on applications," *ISA Trans.*, vol. 17, pp. 3-12, Jan., 1978.

APPENDIX B  
RETROFIT USE OF THE RLU

TABLE OF CONTENTS

SECTION	PAGE
B.1      ASSUMPTIONS. . . . .	96
B.2      OPERATING MODES. . . . .	98
B.3      SUMMARY. . . . .	107

## APPENDIX B

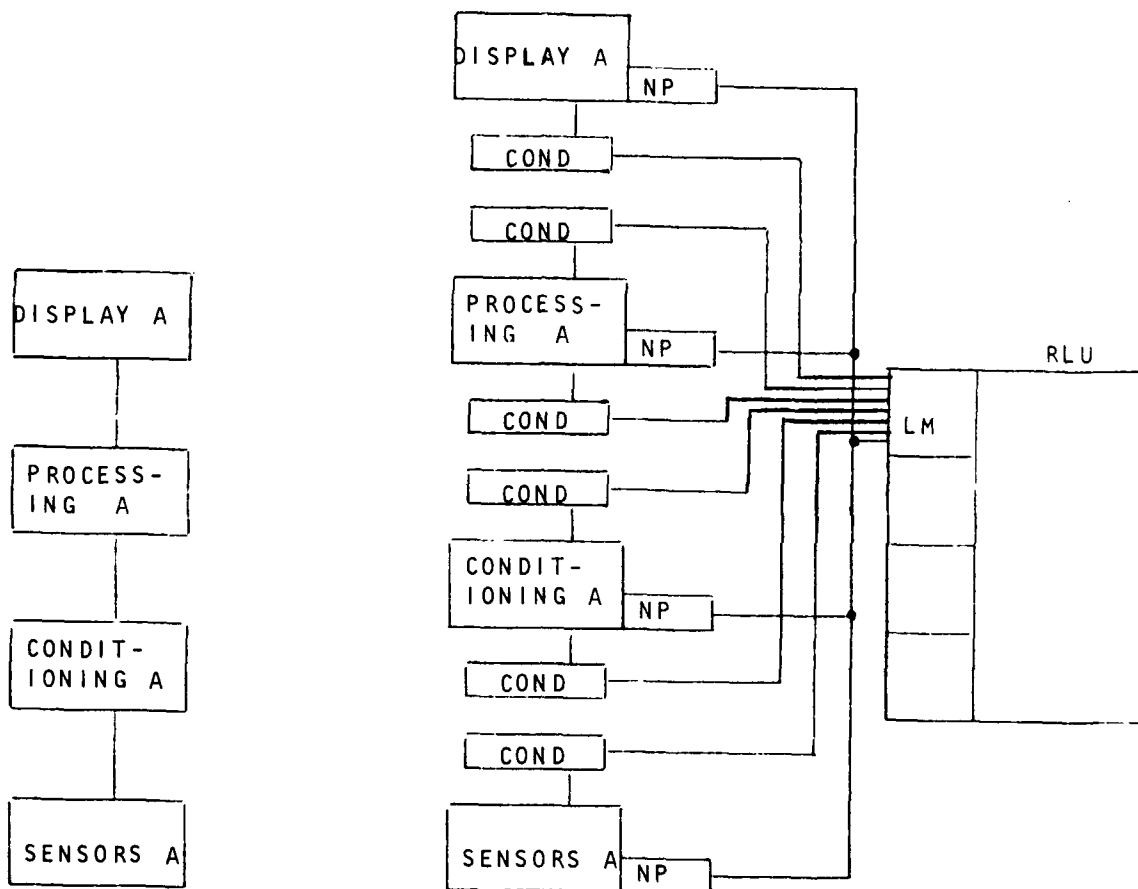
### RETROFIT USE OF THE RLU

This section describes the feasibilities of retrofitting various RLU components into existing avionic subsystems. The authors have made several assumptions concerning these subsystems and these are listed below. Based on these assumptions several possible means of utilizing RLU components are described. Finally, a summary table for retrofit impact is presented. This section does not make specific recommendations for retrofitting RLU's, rather it attempts to consider the ramifications of various retrofit strategies.

#### B.1 ASSUMPTIONS

The ensuing description of retrofitting RLU components into existing avionic subsystems is based on the following assumptions:

1. Each subsystem is self contained. The subsystem architecture is similar to that illustrated in Figure B-1. There are no centralized processing and/or display facilities which are shared by several subsystems.
2. Space is at a premium on board the aircraft



Before Retrofit

After Retrofit

FIGURE B-1 RLU Retrofit to Existing Subsystems

3. Signal type and format used between LRU's are not standardized to any extent. Signal type and format used among the various subsystems are not standardized.
4. It is possible to physically attach a nameplate to each LRU.

## B.2 OPERATING MODES

### B.2.1 Full Use of On-board RLU

The full utilization of an RLU requires that it be permanently mounted in the aircraft, that all interfaced subsystems contain associated nameplates, and that the RLU can effectively interact with the interfaced subsystem. The last requirement is severe in that a significant amount of subsystem information must flow through the RLU. This implies that a fair amount of subsystem processing and signal conditioning is occurring in the RLU and that the RLU has a significant amount of control over the interfaced subsystem. In considering retrofitting of RLU's to previously autonomous subsystems it would seem unlikely that full utilization of an RLU is possible.

### B.2.2 Partial Use of On-board RLU

Partial utilization of on-board RLU's would be

possible for a retrofit operation as depicted in Figure B-1. It is clear from this illustration that a major recabling effort would be required to introduce the RLU into the data paths of a previously closed subsystem. Additionally, special purpose signal conditioning may be required to assure compatibility of RLU and subsystem interfaces. It may be possible for this conditioning to be provided by the ICA in the RLU in some cases, but there will be some subsystems requiring special conditioning.

Once introduced to a subsystem the RLU should be capable of

1. monitoring and recording subsystem failures;
2. doing fairly extensive diagnostics, at least for those subsystem sections of which the RLU has access to both ends (conditioning and processing, see Figure B-2);
3. supporting maintenance by driving the display;  
and
4. performing fault isolation functions.

#### B.2.3 Monitor Mode Use of RLU

The RLU could be easily interfaced to existing subsystems as indicated in Figure B-2. The data paths of the subsystem are unaffected by such a unidirectional tie to the RLU.

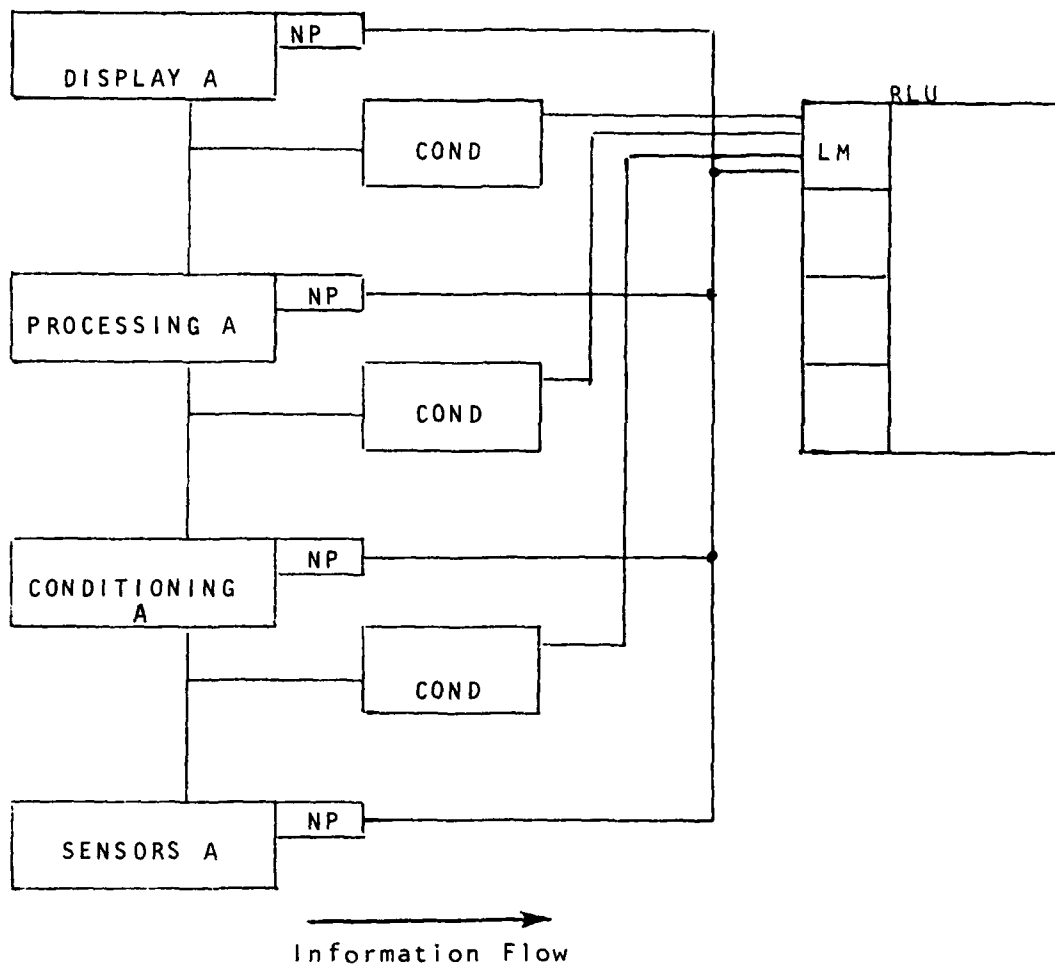


FIGURE B-2 Monitor Mode RLU Interfacing

The interfaced RLU can act in a continuous monitoring mode by comparing signals at various subsystem interfaces. Expected subsystem responses would be available to the RLU through programs and data uploaded from the subsystem LRU nameplates. Detected faults and anomalies can be recorded to the appropriate nameplates. The RLU can also record calibration and environment data to the nameplates.

The use of the RLU illustrated in Figure B-2 is severely handicapped during maintenance by the unidirectional information flow indicated in the figure. Failure summaries would be available to maintenance personnel through the RLU maintenance port.

End-to-end testing could be supported as indicated in Figure B-3, using the radar as an example. This would require some specialized AGE to provide inputs to the subsystem under test.

#### B.2.4 Portable RLU Use

It is likely that the present design of most avionic subsystems will not allow the inclusion of an on-board RLU. Assuming only that nameplates can be attached to existing LRU's it is possible to utilize RLU concepts to help in the preparation, dissemination, and maintenance of repair logs and records.

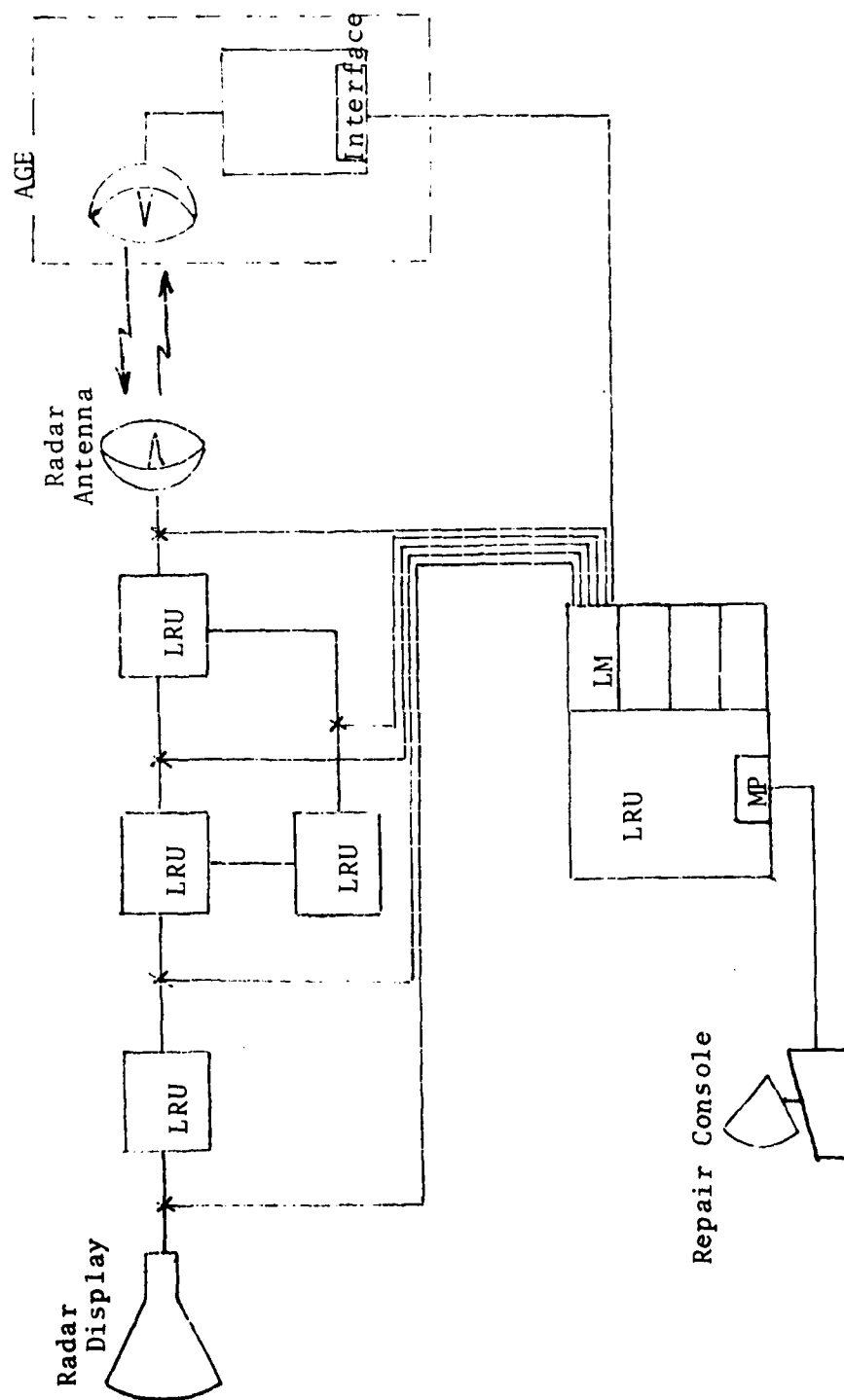


FIGURE B-3 Example of RLU Support of End-to-end Subsystem Testing

Figure B-4 indicates how a portable RLU could be attached to an existing subsystem by use of "piggyback" connectors and additional signal conditioning. The required connector concept is also illustrated in this figure. This configuration allows the connection of the RLU in at least a passive monitoring mode to aid in the repair of faulty subsystems.

The portable RLU connected as described above could serve to call up existing nameplate records from all LRU's. The calibration status and repair history of the LRU's would therefore be available to the maintenance crew. The maintenance crew would also be able to update nameplate records based on current repair and calibration activities. If an LRU were replaced, the repair notes stored on the associated nameplate would be useful to the next maintenance level. The use of the nameplate as a physically attached record store should alleviate many of the current problems related to record collection and updating.

The attachment described above is reliable but will limit the amount of interaction possible between subsystem and RLU. To allow the RLU to communicate with the subsystem requires an extensive recabling effort. The RLU might be able to interact in specific cases by simply overriding normal LRU to LRU data paths. Figure B-5 illustrates one method which might allow an RLU, connected in parallel as

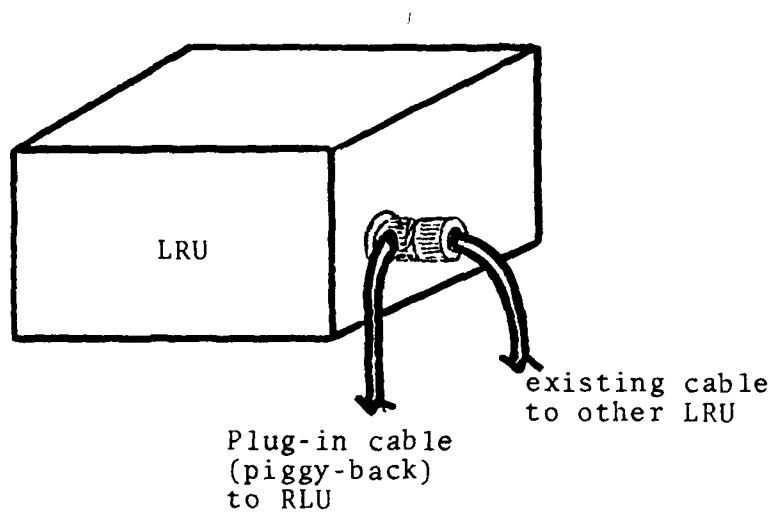


FIGURE B-4 Monitor Mode Piggy-back LRU Connection

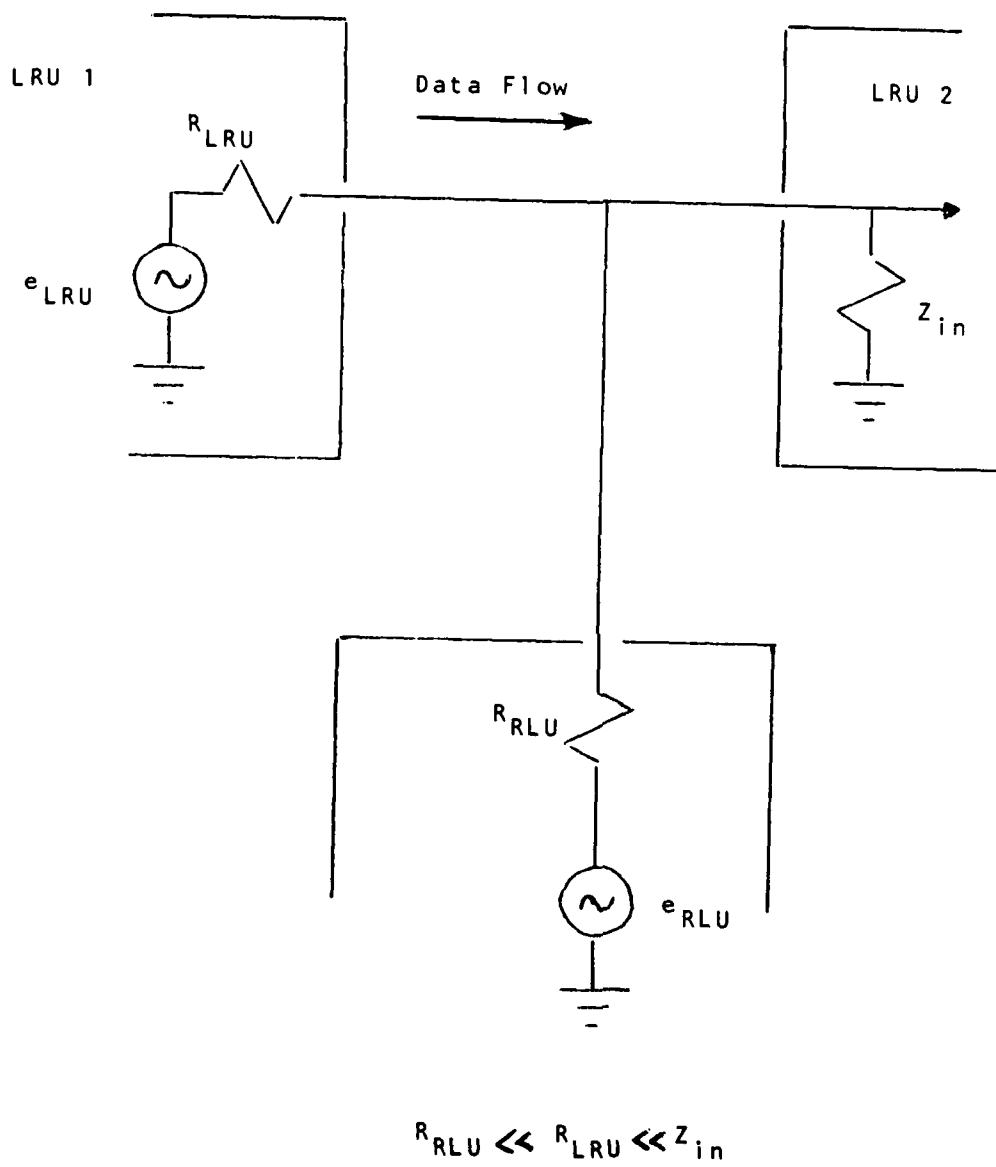


FIGURE B-5 Technique for Inserting RLU Signals into a Data Path

indicated in Figure B-4, to control a data path. Obviously the LRU involved must be capable of supporting such a hookup without being damaged.

#### B.2.5 - Use of Portable RLU with AGE

Some subsystems can utilize a small amount of AGE coupled with a portable RLU connected in a passive manner (see Figure B-2) to act as a powerful repair and maintenance aid. Figure B-3 illustrates the proposed concept for a radar subsystem end-to-end test set up. By using a radar transponder controlled by the RLU coupled with information fed into the RLU Link Module from several points within the subsystem it should be possible to evaluate the subsystem and provide information and instruction to the repair crew. The required AGE would consist of a transponder, antenna, and RLU interface circuitry. Diagnostic routines could be written for the Link Module and stored in the RLU or in some appropriate nameplate. The required AGE can support a similar end-to-end test of other avionic subsystems. For example, an appropriate controllable signal source and interface could be used to test the TACAN radio system, a small controllable temperature source for various thermocouple circuits, etc.

The availability of the 16 independent processors in an RLU could provide for several of the above evaluations to

be run concurrently. Each end-to-end test could be executed in a dedicated Link Module processor with the Link Manager providing communication to and from the repair console as well as providing the overall test coordination.

### B.3 - SUMMARY

The impact of the various retrofits discussed above is summarized in Table B-1.

Partial use of the RLU in a retrofit mode will require extensive cabling changes and interface signal conditioning (see Figure B-1). Also, space for the RLU and nameplates must be found in the aircraft. In this mode it is doubtful that the RLU interfaces added to previously autonomous subsystems would be capable of supporting the level of diagnostic and fault isolation envisioned for subsystems which are designed from the start to take advantage of the testing capabilities of the RLU.

The monitor mode usage of an RLU would require a degree of recabling effort as well as more space for the RLU and the associated nameplates. This type of retrofit would not significantly alter subsystem operation but would form a passive parallel connection to accessible subsystem tie points. Some subsystem-specific signal conditioning would likely be required.

TABLE B-1

## RLU RETROFIT SUMMARY

RLU Use	Cable Changes	Signal Condi- tioning	Space for RLU	Space for NP	Required AGE
I-----I-----I-----I-----I-----I-----I					
I Partial RLU	I Major	Major	yes	yes	I
I Capability	I Effort	Effort			I
I-----I-----I-----I-----I-----I-----I					
I Monitor Mode	I yes	Probable	yes	yes	I
I Use of RLU	I				I
I-----I-----I-----I-----I-----I-----I					
I End-to-End	I yes	yes	Preferred		yes I
I Test Support	I			Preferred	I
I-----I-----I-----I-----I-----I-----I					
I Portable Use	I yes	yes		Preferred	I
I of RLU	I			Probable	I
I-----I-----I-----I-----I-----I-----I					
I Aircraft	I yes	yes		Preferred	I
I Confirmation	I				I
I-----I-----I-----I-----I-----I-----I					

The RLU could support end-to-end tests in either an in-place configuration or in a portable configuration. In terms of maintenance crew convenience, the in-place configuration would be preferable. The RLU would be considered as AGE if used as a portable instrument. The portable RLU would require cabling changes every time it was interfaced to a subsystem, whereas the built-in RLU would be permanently installed in the aircraft and could be accessed through its maintenance port.

The use of an RLU to facilitate the overall functional testing of aircraft subsystems is a logical extension of the end-to-end test implemented with a portable RLU. A single RLU has the capability to support up to 16 simultaneous end-to-end tests and should be of assistance in the routine functional testing and calibration of aircraft subsystems. It should be noted that recabling and AGE would be required to support such testing.

#### REFERENCES

1. C. J. Tavora, J. R. Glover, and G. W. Batten, "Functional Design of the Remote Link Unit," University of Houston, Technical Report No. AFAL-TR-79-1176, August, 1979.
2. L. R. Murphy, A. A. Avizienis, D. A. Rennels, et al., "Fault Tolerant Avionics Systems Architectures Study," Ultrasystems, Incorporated, Technical Report No. AFAL-TR-74-102, June, 1974.
3. W. W. Williams and K. P. Parker, "Testing Logic Networks and Designing for Testability," Computer, IEEE Computer Society, Vol. 12, No. 10, pp. 9-21, October, 1979.
4. "A Designer's Guide to Signature Analysis," Hewlett Packard Application Note 222, Hewlett Packard, 5301 Stevens Creek Blvd., Santa Ana, California 95050.
5. F. Gerkin, et al., "Design-for-Repair Concept Definition," Hughes Aircraft Co., Technical Report No. AFAL-TR-79-1130, June, 1979.
6. P. W. Becker and F. Jensen, Design of Systems and Circuits for Maximum Reliability or Maximum Production Yield, McGraw-Hill Book Co., New York, 1977.
7. A. A. Avizienis, "Fault Tolerance: The Survival Attribute of Digital Systems," IEEE Proceedings, Vol. 66, pp. 1109-1125, October, 1978.
8. J. B. Clary and R. A. Sacane, "Self-testing Computers," Computer, IEEE Computer Society, Vol. 12, No. 10, pp. 49-59, October, 1979.

END

DATE  
FILMED

8-80

DTIC